المعالجات الدقيقة
Microprocessors

Intel80186 - Intel8086 - Z80 - Intel8085
Pentium - Intel80486 - Intel80386 - Intel80286

البرمجة ... والمواجهة ... والتطبيق

تأليف

أ.د. محمد إبراهيم العدوى

أستاذ ورئيس قسم الاتصالات والإلكترونيات

كلية الهندسة - جامعة حلوان

الدار الدولية للاستثمارات الثقافية

المعالجات الدقيقة

Microporocessors

Intel 80186 – Intel 8086 – Intel 8085

Pentium 8 – Intel 80486 – Intel 80386

Intel 80286

البرمجة ... والمواجهة ... والتطبيق

أ.د. محمد إبراهيم العدوى

أستاذ ورئيس قسم الاتصالات والإلكترونيات

كلية الهندسة ـ جامعة حلوان

الدار الدولية للاستثمارات الثقافية

مصر

الطبعة الأولى

2000 م

المعالجات الدقيقة

تأليف

د. محمد إبراهيم العدوى

رقم الإيداع

2000/3149
I.S.B.N 977-282 -076-5

8 إبراهيم العرابى - النزهة الجديدة - مصر الجديدة - القاهـرة - ج.م.ع.

ص.ب: 5599 هليوبوليس غرب/ القاهرة - تليفون: 2957655/2972344 فاكس : 2957655 (00202)

المحتويات

إهداء

إلى اللغة العربية

ظلموك فقالوا إنك لست لغة علوم . . . مع أنك والله مثل الإنجليزية والكثير من اللغات الأوربية (أربعة وعشرون حرفا أو تزيد قليلا) . . . ويتكلم بك عدد لا بأس به مثلها تماما . . . أين أنت من اليابانية التى لها أربعة آلاف حرفا أو تزيد . . . ولكنها قوية بقوة أهلها . . . وعزيزة بعزتها عند أهلها . . . ولكن يكفيك فخرا أنك لغة القرآن الكريم المحفوظ من قبل الله عز وجل . . . وبذلك ضمنت الحفظ والصون إلى الأبد . . . كان من الممكن أن تكونى فى عالم النسيان فى ظل عقد الخواجه التى يعيشها أبنائك الآن لولا حفظك بالقرآن الكريم . . .

حمدا لله

عرض الكتاب

عندما ظهر الترانزيستور فى أواخر الأربعينات الميلادية أحدث ما يشبه الثورة فى مجال الإلكترونيات بحيث أنه أصبح الوحدة الأساسية لبناء أى دائرة إلكترونية. بظهور مكبر العمليات operational amplifier فى منتصف الستينات كدائرة تكاملية رخيصة التكاليف ، صغيرة الحجم ، سلبت الأضواء من الترانزيستور ودخل مكبر العمليات كوحدة أساسية جديدة فى الكثير من التطبيقات والدوائر الإلكترونية . فى منتصف السبعينات تألق نجم الميكروبروسيسور (المعالج) وانتشر استخدامه فى الكثير من الدوائر والأنظمة الإلكترونية وأنظمة التحكم ، حتى أنك تجده الآن فى السيارة يتحكم فى الكثير من متغيراتها ، وفى الصاروخ يتحكم فى توجيهه ، وفى الميكروكومبيوتر يتحكم فى تشغيله ، بل فى لعبة الطفل يتحكم فى الكثير من أدائها .

موضوع المعالجات من الموضوعات التى يصعب جدا أن تجمع أو تنتهى فى مؤلف أو كتاب واحد يرضى جميع القراء وذلك نظرا للتشعبات الكثيرة التى يمكن أن يتشعب إليها هذا الموضوع ، وما من قارئ يقرأ كتابا فى هذا الموضوع إلا ويقول ليت المؤلف أضاف كذا وحذف كذا تبعا لنظرته واهتماماته الخاصة . هناك مثلا من القراء من يهتم بالبرمجة فقط software ولا يهتم كثيرا بموضوعات المواجهة والبناء hardware ، كما أن هناك العكس ، فمن القراء من يهتم كثيرا بالمواجهة والبناء على حساب البرمجة .

لقد راعينا فى هذا الكتاب أن يفى بقدر الإمكان باحتياجات الكثير من القراء ، فهذا الكتاب يقدم للقارئ الكثير من شرائح المعالجات الشائعة الاستخدام ذات 8 و 16 و 32 بت وذلك حتى يجد أى مستخدم لهذه الشرائح ما يفيده وتكون الفرصة متاحة لمن يرغب فى المقارنة بين أكثر من شريحة ويشهد تطور المعالجات عبر الأجيال المختلفة . يحتوى الكتاب أيضا على فصل خاص ببرمجة كل شريحة على حدة من الشرائح 8085 و Z80 و 8086 من خلال شرح ميسر وأمثلة عديدة على أوامر كل شريحة وذلك فى الفصول 4 و 5 و 15 وذلك بعد أن سبقت هذه الفصول بفصل كمقدمة عن المعالجات بصفة عامة وما هو دورها فى الميكروكومبيوتر وهو الفصل الأول ، ثم تلا ذلك فصل خاص بالتركيب الداخلى للميكروبروسيسور بصفة عامة على ضوء الوظائف المطلوبة منه وهو الفصل الثانى حيث انتهى هذا الفصل بعرض التركيب الداخلى للمعالجين 8085 و Z80 ، وبعد ذلك قدم الفصل الثالث عرضا للغات البرمجة وكيفية برمجة المعالج . كتطبيق على الفصول الخمسة الأولى من الكتاب يحتوى الفصل السادس على عملية بناء لمعالج افتراضى hypothetical من البداية (إبتداء من دائرة نصف

المجمع) ثم الارتقاء بهذه الدائرة إلى أن يتم الحصول على وحدة حساب ومنطـق ثم توصيل هذه الوحدة مع مركم accumulator ثم توصيلها مع الذاكـرة وعمـل قائمة أوامر خاصة بهذه الوحدة لبرمجتها . بذلك ينتهى تقريبا جـزء مـهم مـن الكتاب وهو الجزء الخاص بالمقدمة وتركيب المعالجات ذات 8 بت وبرمجتها .
يبدأ بعد ذلك الفصل السابع الذى يحتوى على بعض الأساسيات التى يجب الإلمـلم بها قبل الدخول فى عمليات المواجهة مع المعالج ، مثل عمليات الفصل أو العـزل buffering ومتى نحتاج إليها والمنطق الثلاثى أيضا ولماذا نحتاجه مـع عـرض لبعض الشرائح التى تستخدم فى ذلك . لمواجهة المعالج مع شرائح الذاكرة مثـلا لابد من تهيئة المسارات الثلاثة (العناوين والبيانات والتحكم) والحصول عليها فـى صورة مناسبة لأى عملية مواجهة ، حيث يحتوى الفصـل الثـامن علـى ذلـك بالتفصيل . بعد عملية التهيئة للمسارات الثلاثة فى الفصل الثامن يقـدم الفصـل التاسع كيفية توصيل الذاكرة على المعالج ثم يقدم الفصل العاشر كيفيـة توصيـل المعالج على بوابات أو منافذ الإدخال والإخراج وذلك بالطرق المختلفة . بانتـهاء الفصل الحادى عشر يتم الانتهاء من دراسة الأجزاء الرئيسية اللازمة لبناء دوائر التحكم التى تستخدم المعالج ولذلك فإن الفصل الحادى عشر يقدم مثالا متكـاملا بجزئى البرمجة والبناء لنظام التحكم فى إشارة مرور فى تقاطع ربـاعى وذلـك كمثال يمكن تقليده فى أى تطبيق آخر . البرمجة المتقدمة للمعالج تحتاج لبعـض الموضوعات التى رأينا أن يفرد لكل منـها فصـل خـاص بـها ، مـن هـذه الموضوعات موضوع البرامج الفرعية subroutines والذى أفرد له الفصل الثانى عشر ، ثم موضوع المقاطعة interrupt وقد خصص له الفصل الثالث عشر .
المعالجات ذات 16 بت ويمثلها المعالج intel8086 قد فرضت نفسها على السـوق فترة ليست بالقليلة متمثلة فى الحاسبات XT . لذلك قد أفردنا لها أكثر من فصل ، فالفصل الرابع عشر يقدم تفاصيل التركيب الـهيكلى لـهذا المعـالج ، والفصـل الخامس عشر يقدم تفاصيل برمجة هذا المعالج حيث أن لغة التجميع لهذا المعـالج تعتبر الأساس لكل المعالجات التالية . الفصل السادس عشر يقدم كيفية مواجهـة هذا المعالج مع الدوائر الخارجية مثل الذاكرة وبوابـات الإدخـال والإخـراج . وأخيرا يقدم الفصل السابع عشر فكرة مختصرة ولكننا نعتقد أنـها كافيـة عـن المعالجات 80186 و 80286 و 80386 و 80486 وكذلك عائلة معالجـات بنتيـم الشهيرة فى السوق هذه الأيام والتى تتوالى إصداراتها حيث نفاجأ بإصدار جديـد منها كل ستة شهور تقريبا .
هذا الكتاب ككتاب دراسى text book يمكن تدريسه للمبتدئين فى تعلم موضـوع المعالجات على فصلين دراسيين متعاقبين (ساعتين لكل فصل أو 4 ساعات فـى فصل واحد) حيث يدرس فى الفصل الدراسى الأول المقدمة والفصول الخاصـة ببرمجة المعالج والمعالج الافتراضى وفصل البرامج الفرعيـة وكذلـك مبـادئ

مواجهة المعالجات والتطبيق على ذلك بدراسة فصلى المواجهة مع الذاكرة وبوابات الإدخال والإخراج . فى الفصل الدراسى الثانى يتم تدريس الفصل الخاص المقاطعة واستخدام المعالج فى عمليات التحكم المختلفة أو فى بناء نظام ميكروكومبيوتر بسيط وبعدها يتم الانتقال إلى المعالجات ذات 16 بت لدراسة تركيبها الداخلى وبرمجتها ومواجهتها ثم يتم الانتقال إلى المعالجات الأخرى لأخذ فكرة سريعة عنها ومقارنتها بما سبقها من معالجات .

لابد أن يصاحب هذا المقرر ساعتين للمعمل أسبوعيا يقوم الطالب فيها بتطبيق كل ما تمت دراسته من برامج أو دوائر من خلال بعض التجارب التى توضع بدقة بحيث تتابع الدراسة النظرية للموضوع وتغطى جانبى البرمجة ودوائر مواجهة المعالج .

لكى تتم الفائدة يجب أن يكون القارئ لهذا الكتاب قد درس قدرا كافيا من الإلكترونيات الرقمية ابتداء من البوابات والدوائر المنطقية ، تبسيطها وطرق بناؤها ، ودوائر الجمع adders ، والمشفرات encoders ، والمنتخباتmultiplexer ثم القلابات بأنواعها flip flops ومسجلات الإزاحة shift registers والعدادات counters بأنواعها . كذلك فإنه من الأفضل (وليس بضرورة) أن يكون القارئ قد درس مقررا عن مقدمة الحاسب وألم فيه بموضوعات الخوارزميات Algorithms وخرائط التدفق أو مخططات السير Flow charts والحلقات Loops والقفز Jump ولا يهم أن يكون ذلك بأى لغة من لغات البرمجة ذات المستوى العالى حيث يتساوى فى ذلك الباسيك أو الباسكال أو ال C.

يمكن للقارئ المهتم بأى نوع من أنواع المعالجات التى تم تناولها فى هذا الكتاب أن يتتبع الدراسة والتطبيق على هذا النوع فقط دون عناء كبير ودون أن يكون مضطرا لقراءة الكثير عن المعالجات الأخرى التى لا تهمه إن أراد ذلك ، حيث قد تم مراعاة ذلك فى خلال هذا الكتاب بقدر الإمكان مع العلم أن هناك فصولا تم عرض معظمها بصورة لا تعتمد على المعالج المستخدم مثل فصول مواجهة الذاكرة والإدخال والإخراج .

من الصعوبات التى واجهتنا فى إعداد هذا الكتاب والتى من المتوقع أنها تواجه أى شخص مهتم أو حريص على نشر وترجمة العلوم باللغة العربية هى ترجمة المصطلحات العلمية فى هذا المجال . إننا نقول أنها صعوبات ليس لأن اللغة العربية عاجزة عن إيجاد لفظة عربية تؤدى معنى المصطلح ولكن لأن لفظ المصطلح الأجنبى قد فرض نفسه علينا نحن المشتغلين فى هذا المجال بحيث أصبح من الصعب أن نفلت منه وذلك راجع بالطبع للسبق الذى حققه الناطقون بلغة المصطلح ، ولذلك فقد حاولنا استخدام صورة المصطلح الأجنبية مكتوبة باللغة العربية والإنجليزية مع الترجمة العربية لمعنى المصطلح وذلك حتى نتيح للقارئ فرصة التعرف على المصطلح بلغته الأصلية ، فمثلا كلمة

microprocessor قد شاعت ترجمتها باللغة العربية بالمعالج الدقيق وأحيانا المعالج الصغرى ونحن استخدمنا الترجمة الأولى وفى معظم الأحيان نستخدم كلمة معالج فقط أو حتى كلمة بروسيسور أو ميكروبروسيسور كما يشيع النطق بها حتى تعم الفائدة ولا ننسى المصطلح الأجنبى .

وأخيرا وقبل أن أترك هذا المقام لابد من تقديم كلمات شكر وعرفان لكل من ساعد ولو بالتشجيع فى إنجاز هذا الكتاب الذى أخذ الكثير من الجهد . أخص بالشكر هنا الأستاذ الدكتور فريد عبد العزيز طلبة الأستاذ بكلية الهندسة جامعة عين شمس وعميد كلية التعليم الصناعى بالقاهرة الذى تابع وراجع وأبدى الآراء الصائبة بالذات فى المراحل الأولى من الكتاب . كذلك أتوجه بالشكر إلى زملاء لى فى الكلية التقنية ببريدة بالمملكة العربية السعودية كانوا وراء فكرة التفكير فى وضع هذا الكتاب باللغة العربية وأخص بالذكر منهم المهندس صالح الشبعان وآخرون كثيرون . أتوجه بالشكر أيضا إلى زميلى د.هشام عبد المنعم كشك الذى مارس تدريس هذا الكتاب وكانت له الكثير من الآراء والتوجيهات . كذلك أتوجه بالشكر إلى الكثير من الطلاب بقسم الإلكترونيات بهندسة حلوان الذين مارست معهم تدريس هذه المحتويات لمدة خمس سنوات وكم كنت أتلقى منهم التوجيهات والآراء الصائبة . وأخيرا أخص بالشكر والعرفان الذى لن أنساه لزوجتى وأولادى الذين صبروا على كثيرا وتحملوا منى الكثير على طول فترات انشغالى بإعداد هذا الكتاب .

إن تجربة تأليف كتاب باللغة العربية ليست بالتجربة البسيطة ولكنها تجربة صعبة تحتاج الكثير من الجهد والوقت والتشجيع وإقناع الآخرين بجدوى وفائدة الكتابة باللغة العربية فى المجالات العلمية . ومن هنا أوجه نداء إلى كل الزملاء أعضاء هيئات التدريس فى الجامعات المصرية والعربية ، إذا كان كل منا يفهم فى تخصصه ويجيده فماذا يمنعه من كتابة أفكاره بلغته الأم ؟ والله إن الفائدة لعظيمة من وراء أن يجد الطالب مرجعا باللغة العربية يتساوى مع أكبر المراجع الأجنبية فى الموضوع ، وإن ذلك من واقع تجربتى الشخصية فى هذا المجال على مدى خمس سنوات على الأقل .

المؤلف

الفصل الأول

عصر الميكروبروسيسور

Century of Microprocessor

1-1 عصر الميكروبروسيسور
Century Of Microprocessor

لقد كان للتقدم النشط فى علوم الإلكترونيات والسرعة الهائلة التى يمكن أن تنفذ بها العمليات الحسابية والتعقيد الذى وصلت إليه عملية بناء الشرائح الإلكترونية (الدوائر التكاملية) الأثر الكبير فى جميع نواحى الحياة وعلى الكثير من العلوم المختلفة . كما أن الطفرة الأخيرة التى حدثت فى علوم الحاسبات يرجع الفضل فيها أساسا إلى التقدم فى علوم الإلكترونيات والتكنولوجيا الحديثة والمتطورة فى تصنيع الدوائر التكاملية Integrated circuits ، فما هى الدائرة التكاملية إذن؟ لكى نعرف ما هى الدائرة التكاملية تعال نرجع إلى الوراء فى التاريخ وبالتحديد فى بداية الخمسينات عندما تم اكتشاف الترانزيستور . فى هذا الوقت كانت الدوائر الإلكترونية تبنى أو تصمم باستخدام الصمامات المفرغة التى كان منها ما يكافئ الترانزيستور ومنها ما يكافئ الدايود على سبيل المثال وكان أى صمام من هذه الصمامات عبارة عن اسطوانة زجاجية مفرغة من الهواء يبلغ قطرها حوالى ثلاثة سنتمترات وارتفاعها حوالى سبعة سنتمترات وكانت هذه الصمامات تحتاج لتشغيلها إلى فرق جهد مستمر d.c عالى يبلغ فوق 200 فولت ، ولذلك كانت هذه الصمامات تشع الكثير من الحرارة مما كان يتطلب الكثير من وسائل التبريد لها. لذلك كانت جميع الأجهزة الإلكترونية فى هذا الوقت تعرف بكبر حجمها ، فلك أن تتخيل مثلا أن جهاز حاسب شخصى من أبسط الأجهزة المعروفة الآن ربما كان يشغل حجرتين كاملتين متوسطتى الحجم لو أنه بنى بهذه الصمامات .

باكتشاف أشباه الموصلات وظهور الترانزيستور أخذت أحجام الدوائر الإلكترونية والفراغ الذى تشغله فى الانكماش ، ومنذ ذلك الحين بدأت عجلة التطور فى بناء الدوائر الإلكترونية فى الدوران وأصبح المصممون لا يكتفون ببناء ترانزستور واحد على نفس شريحة شبه الموصل ولكن بدءوا فى وضع أكثر من ترانزيستور على نفس القطعة ، ثم أضافوا لهذا العدد من الترانزستورات بعض المكونات الأخرى مثل المقاومات والمكثفات ، ثم قاموا بتوصيل هذه المكونات مع الترانزستورات الموجودة على نفس الشريحة للحصول على دائرة إلكترونية تؤدى وظيفة معينة ، هذه الدائرة الإلكترونية المبنية على شريحة واحدة لأداء هدف أو وظيفة معينة هى ما يسمى بالدائرة التكاملية . فى بداية الستينات كان كل ما تمكنت منه التكنولوجيا فى ذلك الوقت هو بناء أو تجميع حوالى عشرة ترانزستورات على نفس الشريحة واستخدمت هذه فى بناء دوائر البوابات المنطقية مثل بوابة AND وبوابة OR وبوابة NOT وغيرها وسميت هذه الدوائر بدوائر التكامل الصغير Small Scale Integration (SSI) .

بعد ذلك أخذت تكنولوجيا بناء الدوائر التكاملية فى التطور السريع حيث تمكن المصممون من زيادة كثافة المكونات على نفس الشريحة فظهرت الدوائر ذات التكامل المتوسط Medium Scale Integration (MSI) والتى منها على سبيل المثال دوائر العدادات counters ومسجلات الإزاحة shift registers والكثير من المكبرات التماثلية analog amplifiers المتعددة الأغراض ، ولم يقف الأمر عند هذا الحد بل ظهرت بعد ذلك الدوائر عالية التكامل Large Scale Integration (LSI) والتى منها شرائح الذاكرة memory وشرائح المعالجات بجيليها الأول والثانى ، ولم يقف الأمر عند هذا الحد أيضا بل ظهرت بعد ذلك الدوائر التكاملية الفائقة التكامل Very Large Scale Integration (VLSI) والتى منها بعض شرائح الذاكرة والأجيال الأخيرة من شرائح المعالجات والتى منها الجيل الثالث والرابع ، ولك أن تتخيل الآن أن عدد الترانزستورات على الشريحة الواحدة التى لا تتعدى مساحتها السنتمتر المربع الواحد قد فاق عدة ملايين من الترانزستورات على نفس الشريحة ، فالمعالج بنتيم برو (آخر أجيال المعالجات هذه الأيام) Pentium Pro يحتوى على 5.5 مليون ترانزستور . ويعلم الله وحده ما سيأتى لنا به المستقبل القريب وإلى أين سيصل هذا العقل البشرى ؟ هذه النعمة التى دائما يحاول الإنسان تقليدها ولكنه دائما سيخفق فى تصنيعها!!

لقد كان ظهور المعالج هو السبب فى الطفرة الأخيرة التى ظهرت فى علوم الحاسبات والتى تركت آثارها بالتالى على جميع العلوم الأخرى بل وأوجدت أو أحيت علوما كانت على وشك النسيان بسبب المقدرة المحدودة على معالجة وتخزين البيانات فى تلك الأوقات ، فمن كان يتخيل مثلا أن عملية التحكم فى أنظمة الطاقة الكهربية ابتداء من توليدها وانتهاء بتوزيعها على المشتركين من الممكن أن يلعب الحاسب دورا كبيرا فيها ، جميع المصانع الآن لا تخلو من كومبيوتر يتحكم فى أعقد العمليات الصناعية فيها ، بل إننا إذا انتقلنا إلى المجال الطبى ودخلنا حجرة العمليات لوجدنا الغالبية العظمى من أجهزتها الآن تستخدم الحاسب . إن الحاسب الآن دخل جميع نواحى الحياة فمن كان يتخيل أن يستخدم المعالج فى التحكم فى خلط الهواء بالبنزين بل ومراقبة الكثير من أداء السيارة وأجزائها . إنك من الممكن أن تشترى لعبة لطفلك الآن ليلعب بها فتفاجأ بأن بداخلها معالج يتحكم فيها . إن علم الذكاء الصناعى Artificial intelligence وعلوم الكلام ، (التعرف عليه وتوليده) ، وعلوم الروبوتيات كل هذه تعتبر قليلا من كثير من العلوم التى ما كانت ستصل إلى ما وصلت إليه الآن لولا ظهور المعالج . لذلك فإننى أرى أن يسمى هذا العصر فعلا بعصر (الميكروبروسيسور) المعالجات أسوة بعصر البخار وعصر الكهرباء التى مرت بها البشرية سابقا .

إن نظرة مبسطة على شريحة المعالج ستجد أنها كباقى الشرائح بل والأجهزة الإلكترونية ، لكى تتمكن من استخدامها لابد من قراءة الكتالوج الخاص بها ، والكتالوج الخاص بشرائح المعالج يحتوى عادة على جزأين : **الجزء الأول** يكون خاصا بالتركيب الوظيفى لجميع أجزاء الشريحة ووظيفة جميع أطرافها وشكل الإشارة الناتجة أو المطلوبة على كل طرف من هذه الأطراف . **الجزء الثانى** يحتوى مجموعة أوامر الشريحة والتى بها يمكنك برمجتها وعدد هذه الأوامر يختلف بالطبع من شريحة لأخرى وكذلك صيغ الأوامر تختلف باختلاف الشريحة. لذلك سنحاول من خلال فصول هذا الكتاب أن نقوم بتغطية هذين الجزأين بالإضافة إلى كيفية مواجهة أو توصيل شريحة المعالج على الأجهزة أو الشرائح الخارجية للحصول على نظام الميكروكومبيوتر ذى الكارت الواحد One board microcomputer الذى يستخدم فى الكثير من أغراض التحكم وفى الكثير من التطبيقات العملية ، كل ذلك من خلال دراسة مفصلة ومتأنية للمعالجات ذات 8 بت والمعالجات ذات 16. بعد ذلك سيكون هناك عرضا سريعا للمعالجات 32بت والأجيال الأخيرة منها ودراسة الفرق بينها وبين الأجيال السابقة . قبل أن نترك المقدمة ومحاولة لإتمام النظرة العامة على الموضوع سنعرض فى الجزء القادم لتركيب الميكروكومبيوتر وما يمثله المعالج بداخله .

2-1 أين يقع المعالج فى داخل الميكروكومبيوتر ؟ (Computer Architecture)

إن الفكرة التى يقوم عليها الحاسب ما هى إلا تقليدا لطريقة الإنسان فى حل أى مسألة . أنت مثلا حينما تريد أن تحل مسألة فى الطبيعة أو الإلكترونيات ، ماذا تحتاج ؟ إنك لكى تحل هذه المسألة ستحتاج للآتى :

1. القوانين المهمة لحل هذه المسألة وبالطبع فإنك ستستعين بأحد الكتب التى تحتوى على هذه القوانين .
2. ستحتاج كراسة أو بعض الأوراق لتدوين بعض النتائج الحسابية .
3. ستحتاج أيضا إلى آلة حاسبة لمساعدتك فى إجراء بعض العمليات الحسابية .
4. أخيرا ربما تحتاج إلى آلة طابعة لكتابة تقرير أو وضع الحل فى صورة نهائية لائقة .

هذه الأشياء الأربعة لو تم توفيرها مجتمعة لن تحل المسألة من تلقاء نفسها ولكن لابد من وجود منظم لعملية الحل وهو الشخص نفسه ولابد من وجود خطة للحل أيضا . هذه الأجزاء الأربعة السابقة هى تقريبا ما يتركب منه الحاسب كما سنرى سوى أن خطة الحل وهى البرنامج يتم وضعها للحاسب عن طريق الإنسان بحيث إذا جاء الحل خطأ فلابد أن هناك خطأ فى البرنامج الموضوع للحاسب

بواسطة الشخص المبرمج . أى أن الحاسب لا يحل المسألة من تلقاء نفسه ولكنه يسير على خطة الحل التى وضعتها أنت له مستفيدا فقط بالسرعة الهائلة التى ينفذ بها العمليات . لذلك سنعرض الآن للأجزاء الرئيسية فى الحاسب تاركين للقارئ أمر مقارنة هذه الأجزاء بالأجزاء الأربعة التى ذكرناها سابقا . يتكون الميكروكومبيوتر كما هو موضح فى شكل (1-1) من ثلاثة أجزاء رئيسية هى :

1. الذاكرة Memory.
2. وحدات الإدخال والإخراج Input/Output Ports.
3. وحدة المعالجة المركزية Central Processing Unit.

1-2-1 الذاكرة Memory

الذاكرة ما هى إلا وعاء لحفظ المعلومات لحين الحاجة إليها وهذه المعلومات إما أن تكون بيانات ستكون هناك حاجة إليها فيما بعد أو تكون برنامجا مخزنا فى الذاكرة فى انتظار التنفيذ . إن أى برنامج تكتبه على الحاسب وبأى لغة ولتكن لغة الباسيك BASIC مثلا لابد وأن يوضع أولا فى الذاكرة الأساسية للحاسب ثم يتم استدعاؤه من هناك للتنفيذ عند الأمر بذلك ، أى أن الحاسب لا ينفذ إلا برامج موجودة فى الذاكرة الأساسية فقط . تنقسم الذاكرة عامة إلى قسمين :

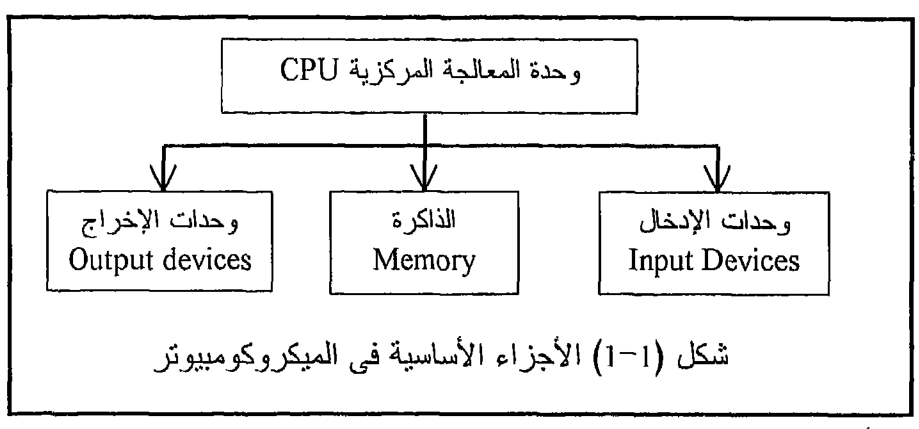
القسم الأول وهو الذاكرة الأساسية للحاسب Main memory وهى التى تخزن فيها البرامج التى تنتظر التنفيذ ، وهذا النوع من الذاكرة يكون عادة من أشباه الموصلات Semiconductors وعلى شرائح تكاملية ويحدد مقدارها على حسب نوع المعالج المستخدم فى الحاسب كما سنرى فيما بعد . الذاكرة الأساسية للحاسب تنقسم بدورها إلى جزأين :

الأول : وهو ما يسمى بذاكرة القراءة فقط Read Only Memory , ROM وهذه الذاكرة أيضا تكون عبارة عن شرائح من أشباه الموصلات التى تم تسجيل محتوياتها بواسطة الصانع نفسه وعادة تحتوى الثوابت والبرامج المهمة لتشغيل نظام الحاسب والتى لا تضيع بانقطاع مصدر الطاقة عنها .

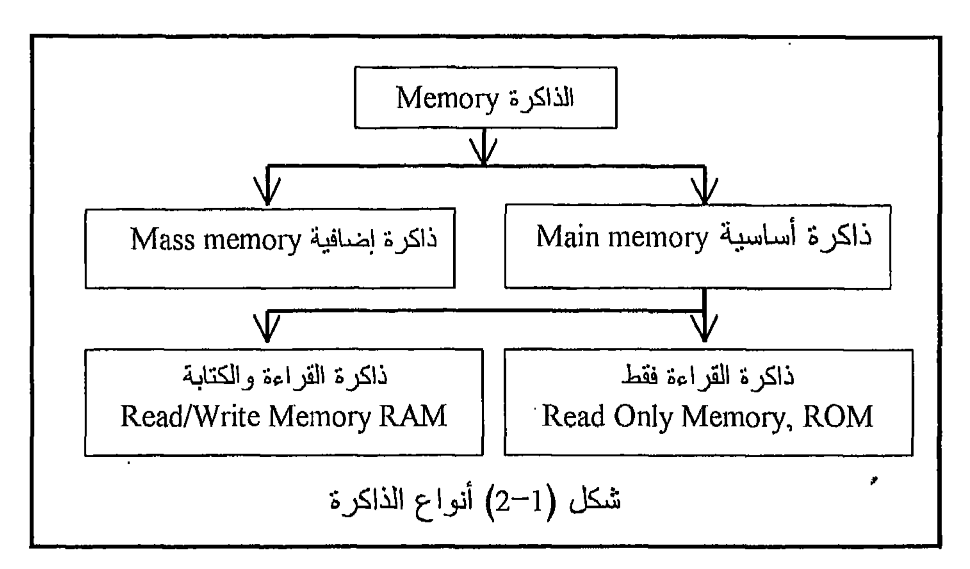
الثانى : وهو ذاكرة القراءة والكتابة Read/Write Memory ولقد تم العرف على تسمية هذا النوع من الذاكرة بذاكرة الاتصال العشوائى Random Access Memory, RAM وهى الذاكرة التى تحتوى البيانات والبرامج التى فى انتظار التنفيذ كما ذكرنا من قبل وهذه الذاكرة تفقد محتوياتها بانقطاع مصدر الطاقة .

القسم الثانى من أقسام الذاكرة هو الذاكرة الإضافية أو Mass Memory وهى الذاكرة التى تستخدم لتخزين البيانات أو البرامج لفترات طويلة وعادة فإن هذه الذاكرة تكون مغناطيسية مثل الأقراص Floppy disks والشرائط Tapes وهناك

أيضا الأقراص الصلبة Hard Disks . هذا القسم من الذاكرة لا دخل للمعالج فـي تحديد كميته ولكن كميته تحدد على حسب رغبة المستخدم ومـا وصلـت إليـه التكنولوجيا فى هذا المجال . شكل (1-2) يبين رسما توضيحيا لأقسام الذاكرة فى الحاسب التى سبق الحديث عنها .



شكل (1-1) الأجزاء الأساسية فى الميكروكومبيوتر



شكل (1-2) أنواع الذاكرة

1-2-2 وحدات الإدخال والإخراج Input/output Ports

وحدات الإدخال هى الوسائل التى يتم بها تكييف المعلومات لتكون فـي صـورة مناسبة يستطيع البروسيسور التعامل معها ، ومثال ذلك لوحة المفاتيح التى تحول أى زرار تقوم بضغطه إلى إشارات كهربية وشفرات يقبلها المعالج . يجـب أن نفرق هنا بين بوابة الإدخال ووحدة الإدخال حيث بوابة الإدخال يتم من خلالـها

إدخال المعلومات التى تم تجهيزها بواسطة وحدة الإدخال إلى المعالج كما سنرى بالتفصيل فى فصول الكتاب القادمة .
وأما وحدات الإخراج فهى الوسائل التى يتم بها إظهار المعلومات الخارجة من المعالج ، ومثال ذلك الشاشة التى ما هى إلا وسيلة ضوئية لإظهار المعلومات التى تخرج من المعالج ، بالطبع فإن هذه الشاشة تكون متصلة بأحد بوابات الإخراج ، ولذلك يجب أن نفرق هنا بين بوابات الإخراج ووحدات الإخراج حيث بوابة الإخراج يتم من خلالها إخراج المعلومة من البروسيسور إلى وحدة الإخراج التى تتعامل مع هذه المعلومات بوسائل مختلفة كما سنرى بالتفصيل .

3-2-1 وحدة المعالجة المركزية
Central Processing Unit, cpu

الوظيفة الأساسية لوحدة المعالجة المركزية هى تنفيذ البرامج عن طريق إحضار الأوامر من الذاكرة الواحد بعد الآخر ثم تنفيذها بنفس التتابع ، فمثلا يتم إحضار الأمر الأول ثم ينفذ وبعد ذلك يحضر الأمر الثانى وينفذ فالأمر الثالث وينفذ وهكذا إلى أن تصل إلى نهاية البرنامج . بعض هذه الأوامر تحتاج لبيانات من أماكن أخرى فى الذاكرة يتم إحضارها ، وبعضها يحتاج لبيانات من بوابات إدخال يتم إحضارها أيضا ، والبعض الآخر من الأوامر يتطلب كتابة أو تسجيل بعض البيانات إما فى الذاكرة أو فى وحدات إخراج ، كل ذلك وأكثر تقوم به وحدة المعالجة المركزية . فى معظم أنظمة الميكروكومبيوتر الشخصية تكون وحدة المعالجة المركزية هى شريحة أو أكثر من شرائح الميكروبروسيسور أو المعالج الدقيق كما سمى بالعربية والذى هو موضوع دراسة هذا الكتاب .

3-1 ماذا تعنى هذه الألفاظ ؟

نسمع هذه الأيام الكثير من الألفاظ والتى لا نعرف مدلولها الدقيق ولا ماذا تعنى هذه الألفاظ ؟ لذلك سنقدم فى هذا الجزء بعض هذه الألفاظ مع شرح بسيط لمدلولها والاستعانة ببعض الأمثلة إن أمكن .

1-3-1 الميكروكومبيوتر والميكروبروسيسور

لقد رأينا فى هذا الفصل كيف أن كلمة ميكروبروسيسور تعنى تلك الشريحة ذات الأطراف المتعددة والقادرة على تنفيذ مجموعة من الأوامر المحددة بحيث ينفذ كل أمر عند إعطاء الشفرة الخاصة به . كلما تعددت هذه الأوامر ، وكلما كان المعالج أسرع فى تنفيذ هذه الأوامر ، وكلما كان المعالج أسهل فى عمليات

المواجهة مع الدوائر المحيطة كلما كان المعالج أفضل . فى الكثير من الأحيـــان يستخدم لفظ "بروسيسور" فقط للدلالة على نفس الشيء ، ونحن فى هذا الكتـــاب سنستخدم أى واحد من اللفظين "بروسيسور" أو "ميكروبروسيسور" أو المــرادف العربى لهما وهى كلمة "المعالج" نظرا لشيوع استخدام كل هذه الألفاظ .
أما الميكروكومبيوتر فقد رأينا سابقا أنه ذلك الجهاز الذى يتكـــون مـــن بعــض الأجزاء الثانوية مثل الذاكرة ووحدات الإدخال والإخراج وجزء أساســـى وهـــو المعالج . أى أن المعالج يعتـــبر جـــزءا أساســـيا بـــل هـــو أهـــم جـــزء فـــى الميكروكومبيوتر . عند ذكر كلمة ميكروكومبيوتر يتبادر إلى ذهننا فـــورا تلـــك المجموعة المكونة من شاشة للعرض ولوحة مفاتيح وطابعة وغـــير ذلـــك مـــن الأجهزة ، ولكن فى الحقيقة فإن هذا هو أحد أشكال الميكروكومبيوتر موضوعـــا فى صورة تسهل عملية التعامل معه وبرمجته حتى من غير المختصيـــن الذيـــن يتعاملون معه بغرض البرمجة فقط باستخدام اللغات المعروفة . هنـــاك صـــورا أخرى للميكروكومبيوتر غير هذه الصورة المألوفة مثـــل "الميكروكومبيوتـــر ذو الكارت الواحد" مثلا وهو عبارة عن كارت واحد عليه شريحة المعالج وشـــريحة ذاكرة والقليل من بوابات الإدخال والإخراج ، كل ذلك مبنى على كارت واحـــد لأداء غرض معين مثل التحكم فى أى عملية صناعية كما سنرى فى هذا الكتاب. بل إن هناك صورة أخرى للميكروكومبيوتر وهى الميكروكومبيوتر على شـــريحة واحدة ، نعم شريحة واحدة تحتوى معـــالج وبعـــض الذاكـــرة (RAM و ROM) وبعض بوابات الإدخال والإخراج . بل إن هناك بعض شرائح الميكروكومبيوتـــر التى تحتوى الأكثر من ذلك مثل المحولات مـــن تمـــاثلى إلـــى رقمـــى (A/D) والمحولات من رقمى إلى تمـــاثلى (D/A) والمؤقتـــات (Timers) والمرشـــحات الرقمية (Digital Filters) وغير ذلك وعادة ما يطلـــق علـــى هـــذه الشـــرائح الحاكمات الدقيقة Microcontrollers .

2-3-1 البرمجة والبناء Software and Hardware

يكون التعامل مع المعالج فى العادة بوسيلة من اثنتين لا غنى لواحدة منهما عـــن الأخرى :
الوسيلة الأولى : هى برمجة المعالج وهو ما يسمى software وعادة مـــا تكـــون البرمجة بلغة الماكينة الخاصة بالبروسيسور الذى تتعامل معـــه حيـــث أن كـــل بروسيسور له لغة ماكينة خاصة به كما سنرى فى هذا الكتاب .
الوسيلة الثانية: هى البنـــاء hardware وتشـــتمل علـــى مواجهـــة أو توصيـــل البروسيسور على الدوائر المحيطة مثل الذاكرة وبوابـــات الإدخـــال والإخـــراج واستخدام البروسيسور فى التطبيقات المختلفة مثل دوائـــر التحكـــم مثـــلا . إن المتعامل مع المعالج لابد وأن يكون ملما بكلتا الوسيلتين الســـابقتين ، البرمجـــة

والبناء ، وإن اختلفت نسبة إلمامه بأى واحدة منهما ، فإنه من الصعب لشخص ما أن يتعامل مع البروسيسور بغرض البرمجة فقط لأن ذلك سيعتبر إهدارا لوقته حيث إنه إذا كان يريد غرض البرمجة فقط فيمكنه استخدام أى واحدة من اللغات المعروفة والتى يكون تعلمها أسهل بكثير من تعلم برمجة البروسيسور بلغة الماكينة أو الأسمبلى . أى أن من يتعامل مع البروسيسور فإنه غالبا يتعامل معه بغرض استخدامه كعنصر فعال فى دائرة تحكم لمميزاته المختلفة ، لذلك لابد لهذا الشخص أن يتعلم برمجة البروسيسور ومواجهته واستخدامه فى الأغراض المختلفة . وعادة تطلق كلمة Software على البرمجة بأى لغة من اللغات وأما كلمة Hardware فتطلق على عمليات البناء والتركيب الإلكترونى بأى شكل ولأى غرض .

1-3-3 الأمر والبرنامج Instruction and Program

أعطنى القلم ! ! هذا أمر Instruction ..... وأما

1. خذ المفتاح
2. افتح درج المكتب
3. خذ القلم
4. اكتب "أنا أتعلم البرمجة"
5. أعد القلم
6. أغلق الدرج
7. أعطنى المفتاح

!!!

فهذا برنامج مكون من سبعة أوامر .

أى أن الأمر هو وحدة بناء البرنامج ، بينما البرنامج يتكون من مجموعة من الأوامر التى لو تم تنفيذها بالتتابع المحدد فإنها تؤدى هدف أو وظيفة معينة . أى برنامج لابد وأن ينفذ بالتتابع الذى كتب به وهذا هو ما يفعله المعالج بحيث إذا وجد هناك خطأ ما فى النتيجة فإن ذلك يرجع إلى البرنامج وليس إلى المعالج لأن المعالج ما هو إلا ماكينة مطيعة تنفذ ما يطلب منها حتى ولو كان خطأ طالما أن ذلك فى حدود إمكانياتها .

1-3-4 لغات البرمجة Programming Languages

تخيل أن لدينا ماكينة وهذه الماكينة لها 3 مفاتيح تشغيل ، كل مفتاح يكون إما ON أو OFF ، وعلى ضوء حالة المفاتيح الثلاثة يمكن لهذه الماكينة أن تنفذ عملية معينة . جدول 1-1 يبين جميع الحالات الممكنة للمفاتيح الثلاثة والعملية المقابلة لكل حالة . هذه العمليات التى تستطيع الماكينة تنفيذها هى مجموعة

الأوامر الخاصة بهذه الماكينة وعلى ضوء هذه العمليات وبالاستعانة بالجدول 1-1 فإننا نستطيع إعادة كتابة البرنامج الموجود فى الجزء السابق فى صورة شفرات تستطيع هذه الماكينة تنفيذها ، جدول 1-2 يبين هذا البرنامج وقد أعيدت كتابته وفى مقابل كل أمر الشفرة الخاصة به . مجموعة الأوامر المكتوبة بالوحايد والأصفار فى جدول 1-2 هى البرنامج مكتوبا بلغة الماكينة لهذه الآلة التى نستخدمها . لذلك فإنه عامة وكما سنرى فى هذا الكتاب فإن لغة الماكينة تكون عبارة عن شفرات ثنائية (وحايد وأصفار) تدخل إلى البروسيسور فينفذها وهى الصورة الوحيدة التى يمكن التعامل بها مع أى بروسيسور ، وكما سنرى أيضا فإن لكل معالج لغة الماكينة الخاصة به .

S2 S1 S0	العملية المنفذة
0 0 0	خذ (يأخذ أى شيء يعطى له فى يده)
0 0 1	افتح (يستخدم ما معه لفتح ما يحدد له)
0 1 0	أكتب (يستخدم ما معه للكتابة)
0 1 1	ضع (ضع ما معك فى المكان المحدد)
1 0 0	أغلق (يغلق ما يحدد له)
1 0 1	أعطنى (يعطى ما معه)
1 1 0	اذهب (يذهب إلى مكان محدد)
1 1 1	تكرار (يكرر ما يحدد له عدد من المرات)

جدول 1-1 مجموعة أوامر ماكينة افتراضية

مما سبق يتضح لنا أن الكتابة بلغة الماكينة ليست سهلة وأصعب منها اكتشاف الأخطاء فيها ، لذلك فإنه يمكن الكتابة بالأوامر خذ وافتح واكتب وهكذا ولكن فى هذه الحالة بدلا من إدخال الأوامر على الماكينة مباشرة فإننا ندخلها على مترجم يقوم بترجمة الكلمات خذ واكتب إلى شفراتها الثنائية ثم يدخلها على الماكينة . هذا المترجم يسمى أسمبلر والأوامر المكتوبة بلغة أكتب وخذ سنسميها لغة الأسمبلى . هذا هو الوضع تماما فى حالة المعالج حيث أن كل بروسيسور له لغة ماكينة وأسمبلر ولغة أسمبلى خاصة به . لذلك فإن لغة الأسمبلى هى أقرب اللغات من مستوى لغة الماكينة . الآن ما رأيك لو استطعنا تصميم مترجم آخر يأخذ أوامر أكثر تعقيدا مثل "أفتح الدرج وخذ القلم وأكتب" حيث سيقوم المترجم بترجمة ذلك الأمر إلى عدد من أوامر لغة الماكينة وليس إلى أمر واحد كما فى حالة الأسمبلر ، مثل هذه اللغة التى يكون كل أمر فيها يؤدى وظيفة أكثر من أمر من أوامر لغة الأسمبلى تسمى باللغات ذات المستوى العالى High level

language ومنها على سبيل المثال لا الحصر لغة الباسيك والفورتران والبسكال وغيرها كثير . إن المترجم الذى يقوم بالترجمة من لغة ذات مستوى عال إلى لغة ماكينة يسمى المؤلف أو المفسر أو المترجم أو compiler .

الأمـــر	حالة المفاتيح S0 S1 S2
خذ (المفتاح)	0 0 0
افتح (درج المكتب)	0 0 1
خذ (القلم)	0 0 0
اكتب (أنا أتعلم البرمجة)	0 1 0
ضع (القلم)	0 1 1
أغلق (الدرج)	1 0 0
أعطنى (المفتاح)	1 0 1

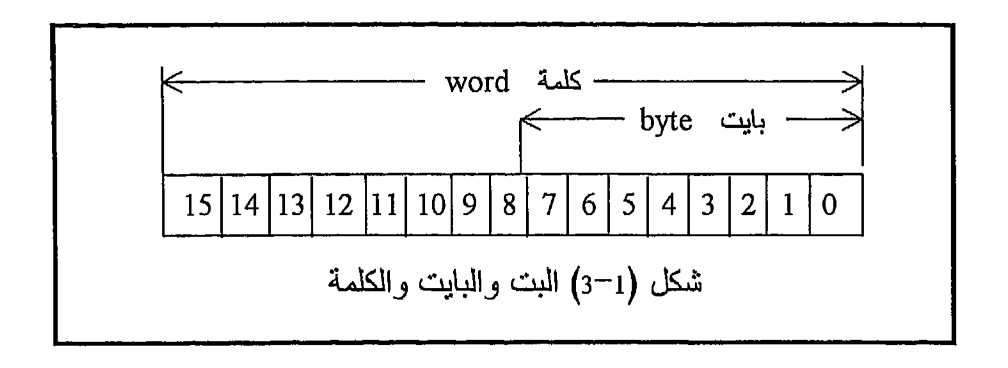
جدول 1-2 برنامج مكتوب بلغة الماكينة الافتراضية

1-3-5 البت bit والبايت byte

البت هى الخانة فى نظام العد الثنائى ، فكما أن العدد العشرى 325 مثلا مكون من ثلاث خانات فإن العدد الثنائى 11001 مكون من خمس خانات أو خمس بتات 5 bits حيث كل خانة تحتوى على واحد أو صفر . عمليا وكما نعلم من الإلكترونيات الرقمية فإن البت تكون عبارة عن قلاب flip flop أو أحيانا تسمى ماسك latch يتم وضع خرجه على القيمة واحد أو صفر . كل ثمانية بتات تكون فيما بينها ما يسمى بالبايت byte ، والبايت هى وحدة تقدير الذاكرة فنقول مثلا أن هذا الحاسب ملحق به ذاكرة مقدارها 64 كيلو بايت أى 65536 بايت حيث أن الواحد كيلو بايت يساوى 1024 بايت كما سنعرف بالتفصيل فيما بعد . شكل (1-3) يبين الفرق بين البت والبايت . كل اثنين بايت تكونان ما يسمى بالكلمة word وعلى ذلك فإن الكلمة word تتكون من 16 بت أو اثنين بايت كما هو موضح فى شكل (1-3) أيضا .

1-4 تمارين

1. وضح بالرسم الصندوقى أجزاء الميكروكومبيوتر واشرح باختصار وظيفة كل جزء ؟



شكل (1–3) البت والبايت والكلمة

2. طابق بين الأجزاء التى شرحتها فى السؤال السابق وما تحتاج إليه من أشـياء لحل مسألة فى الرياضيات مثلا كما هو موضح فى الفصل ؟
3. اشرح أنواع الذاكرة وخصائص كل نوع ؟
4. دليل التليفون ، هل تطابقه مع RAM أم ROM ؟
5. شريط الكاسيت ، هل تطابقه مع RAM أم ROM ؟
6. القرص الممغنط floppy disk ، هل هو RAM أم ROM ؟ وإذا صنفته علـى أنه RAM فهل هى أساسية أم إضافية ؟
7. الفأرة mouse ، هل هى وحدة إدخال أم وحدة إخراج ؟
8. الطابعة ، هل هى وحدة إدخال أم وحدة إخراج ؟
9. المساح scanner ، هل هو وحدة إدخال أم وحدة إخراج ؟
10. الراسم plotter ، هل هو وحدة إدخال أم وحدة إخراج ؟
11. بافتراض أن لك تعاملات مسبقة مع الحاسب الآلى ، فما نوع المعالج الموجود فى الحاسب الذى تتعامل معه ؟
12. ما هو الفرق بين المعالج والميكروكومبيوتر ؟
13. إذا شبهنا الميكروكومبيوتر بالسيارة ، فماذا يمثل المعالج فى هذه السيارة ؟
14. ارسم مخطط سير flow chart لنشاطك اليومى من الصباح حتى النوم فى أيام العمل وأيام العطلات ؟

الفصل الثانى

البناء المعمارى للمعالج

Microprocessor Architecture

2-1 مقدمة

فى هذا الفصل سيتم عرض المهام الأساسية المطلوبة من أى معالج بصفة عامة وعلى ضوء هذه المهام سنعرض الوظائف الأساسية للمكونات العامة لأى شريحة معالج ، ثم نقدم التركيب التفصيلى لاثنين من الشرائح المعروفة وهى الشريحة Intel8085 والشريحة Z80 على أساس أن هذه هى أكثر شرائح الجيل الثانى استخداما وبعد ذلك سنترك للقارئ رؤية مدى ملائمة هذا التركيب للمهام المطروحة . وسوف نعرض التركيب التفصيلى لهذين المعالجين بصورة مختصرة وسريعة حتى يتمكن أى قارئ من مراجعة المكونات الأساسية لهما حتى ولو كان لا ينوى التدريب إلا على أحدهما . لذلك فإننا ننصح بقراءة هذا الفصل بأكمله من مستخدمى هذين المعالجين بالذات أو المعالجات التى نقوم بعرضها فى هذا الكتاب . ولقد رأينا فى الفصل السابق (عصر المعالجات) أن وظيفة المعالج الأساسية هى إحضار الأوامر من الذاكرة وتنفيذها الواحد بعد الآخر ، ولذلك فإن تركيبه الداخلى يجب أن يناسب هذه المهمة أو الوظيفة .

2-2 المهام الأساسية المطلوبة من المعالج

1. يجب أن يكون المعالج قادرا على إحضار معلومات من الذاكرة (هذه المعلومات قد تكون بيانات يحتاجها فى عملية تنفيذ الأوامر أو قد تكون الأوامر نفسها) .
2. يجب أن يحتوى المعالج على مكان مناسب بداخله لحفظ هذه المعلومات التى أحضرها لحين الحاجة إليها أو تنفيذها إذا كانت من الأوامر .
3. لابد أن يكون هناك أكثر من مكان بداخله بحيث يمكن نقل المعلومات فيما بين هذه الأماكن حيث تحتاج بعض الأوامر لذلك عند تنفيذها .
4. يجب أن تكون لديه الوسائل المناسبة لإدخال معلومات من بوابات إدخال حتى يتسنى لنا قراءة لوحة مفاتيح أو إدخال درجة حرارة مثلا تمهيدا لمعالجتها رقميا .
5. يجب أن تكون لديه المقدرة على إجراء بعض العمليات الحسابية والمنطقية على البيانات التى أحضرها . العمليات الحسابية الأساسية هى الجمع والطرح والعمليات المنطقية الأساسية مثل AND و OR و NOT .
6. المقدرة على إرسال بيانات إلى الذاكرة وتسجيلها فيها من المهام الأساسية للمعالج .

7. المقدرة على إرسال بيانات إلى وحدات إخراج من خلال بوابات إخراج حتى يتسنى لنا قراءة هذه المعلومات على شاشة أو إخراج بيانات نتحكم بها فى سرعة موتور مثلا .

كانت هذه هى المهام الأساسية للمعالج والتى يجب أن يحققها تركيبه الداخلى ومجموعة أوامره كما سنرى . سنبدأ فيما يلى الحديث عن مجموعة المسجلات والعدادات التى يشتمل عليها أى معالج حيث أنه من الضرورى لأى مستخدم للمعالج أن يعرف خصائص تلك المسجلات ووظائفها .

2-3 أجزاء المعالج الأساسية

جميع شرائح المعالجات تتركب من ثلاثة أجزاء رئيسية وهى :

1. مجموعة مسجلات وعدادات .
2. وحدة الحساب والمنطق ALU .
3. وحدة التزامن Clock .

بالنسبة لوحدة الحساب والمنطق سوف نرجئ الحديث عنها الآن حيث سيتم إفراد فصل قادم خاص بها (الفصل السادس) وأما وحدة التزامن فسوف يتم الحديث عنها أيضا لاحقا وفى معرض الكلام عن وظيفة كل طرف من أطراف شريحة المعالج . أما مجموعة المسجلات والعدادات ووظيفة كل منها فسوف تكون الموضوع الأساسى فى الجزء القادم .

2-4 المسجلات والعدادات فى شريحة المعالج

تستخدم المسجلات للتخزين المؤقت للمعلومات فى صورة خانات ثنائية فى داخل شريحة المعالج لحين الحاجة إليها . إن أى مسجل إزاحة يمكن تصميمه ليكون قادرا على أداء الوظائف التالية :

1. إدخال المعلومات بالتوالى وإخراجها بالتوالى (سواء من الشمال لليمين أو من اليمين للشمال) .
2. دوران المعلومات فى أى اتجاه وعكسه .
3. إدخال المعلومات بالتوازى وإخراجها بالتوازى .
4. إدخال المعلومات توالي من أى اتجاه وإخراجها توازي أو العكس .

المسجلات داخل المعالج يمكن النظر إليها على أنها واحد من نوعين ، الأول هو مسجلات عامة الأغراض general purpose registers وهذه تستخدم فى الكثير من الأغراض وتؤدى أكثر من وظيفة وعادة تكون هذه المسجلات متاحة

للمستخدم لكى يتعامل معها ، إما أن يسجل فيها أو يقرأ منها ، وأما النوع الثانى فهو مسجلات خاصة الأغراض dedicated registers وهذه مسجلات موجودة لأداء غرض أو وظيفة واحدة لا تحيد عنها وليس للمستخدم أى وسيلة للتحكم فيها سواء بالقراءة منها أو الكتابة فيها .
أما العدادات counters فتستخدم عادة لعد النبضات الداخلة إليها ويمكن توظيف هذه العدادات لكى تقوم بعملية العد إما تصاعديا أو تنازليا مع ملاحظة أن خرج العدادات يكون دائما توازي . سنعرض فيما يأتى بشكل عام لوظيفة كل مسجل من المسجلات الرئيسية لشريحة المعالج وذلك دون تخصيص معالج معين لأن ذلك مطبق على جميع المعالجات التى سنتعامل معها فى هذا الكتاب .

2-4-1 مسجل التراكم Accumulator, A

أى مسجل يمكن النظر إليه على أنه بايت من بايتات الذاكرة وهذه البايت موجودة داخل شريحة المعالج وعادة تكون هناك حرية أكثر فى التعامل مع البيانات الموجودة داخل هذه المسجلات عن البيانات الموجودة فى الذاكرة . من هذه المسجلات ما يسمى بمسجل التراكم Accumulator أو المركم . يعتبر مسجل التراكم ، وعادة يرمز له بالرمز A ، من أكثر مسجلات المعالج عملا ولذلك فإنه يمكننا النظر إليه على أنه سكرتيرا لشريحة المعالج . إن أى عملية حسابية أو منطقية يقوم بها المعالج لابد وأن يكون مسجل التراكم طرفا فيها ، فمثلا لو أردت أن تجمع أى رقمين فإن واحدا منهما لابد وأن يوضع فى مسجل التراكم وأما الرقم الآخر فيوضع فى أى مسجل آخر أو حتى فى الذاكرة ، ليس هذا فقط بل إن نتيجة أى عملية حسابية أو منطقية لا توضع إلا فى مسجل التراكم ومنه يمكن نقلها لأى مكان آخر وذلك فى المعالجات 8 بت . هناك مهمة أخرى أيضا لهذا المسجل وهى أن أى عملية إدخال أو إخراج من خلال بوابات الإدخال أو الإخراج عادة تكون من خلال هذا المسجل . أى أن المعلومة توضع فى مسجل التراكم أولا ثم يتم إخراجها إلى بوابة الإخراج ، أو إذا كانت المعلومة قادمة من بوابة إدخال فإنها توضع أولا فى مسجل التراكم ثم يتم نقلها منه لأى مكان آخر فى داخل البروسيسور أو خارجه .
إذن ما رأيك الآن فى تسميته بسكرتير المعالج ؟ إن عدد البتات (الخانات) bits الموجودة فى مسجل التراكم دائما يساوى عدد خطوط مسار البيانات data bus ومن الممكن فى بعض المعالجات أن يكون هناك أكثر من مسجل تراكم واحد كما سنرى . بعض هذه الوظائف الخاصة بالمركم سيتم الاستغناء عنها فى المعالجات 16 بت كما سنرى .

2-4-2 عداد البرنامج Program Counter, PC

كما علمنا فإن مهمة المعالج الأساسية هى إحضار الأوامر من الذاكرة الواحد بعد الأخر ثم تنفيذها ، ولذلك فإنه لابد لهذه المهمة من تحديد للأماكن التى تحتوى هذه الأوامر فى الذاكرة . يحتوى عداد البرنامج دائما علي عنوان المكان فى الذاكرة الذى يحتوى الأمر الذى عليه الدور فى التنفيذ ، وكلما تم إحضار أى أمر من الذاكرة وقبل أن يتم تنفيذه فإن عداد البرنامج تتغير محتوياته بحيث تشير إلى عنوان الأمر القادم فى التنفيذ . تذكر أيضا أنه حتى لو حدث قفز من مكان فى البرنامج إلى مكان آخر فإن وحدة التحكم داخل المعالج تضع عنوان الأمر الذى سيتم القفز إليه فى عداد البرنامج حتى يصبح هو الأمر الذى عليه الدور فى التنفيذ وتنتقل عملية تنفيذ البرنامج إلى هناك . عدد بتات هذا العداد دائما تساوى عدد بتات مسار العناوين address bus وهذا منطقى جدا حتى يتمكن المعالج من إحضار الأوامر مهما كانت فى أى مكان فى الذاكرة سواء فى أولها أو فى آخرها ، لاحظ أن كمية الذاكرة التى يمكن أن يتعامل معها المعالج تتوقف على عدد البتات أو الخطوط فى مسار العناوين كما سنرى فيما بعد . إذا نظرنا إلى عداد البرنامج على أنه مسجل يحتوى عنوان الأمر الذى عليه الدور فى التنفيذ فإننا سنصنفه على أنه من المسجلات ذات الأغراض الخاصة dedicated register ، لاحظ أيضا أنك كمبرمج لا تستطيع التحكم فى محتويات هذا العداد .

2-4-3 مسجل وفاكك شفرة الأوامر

Instruction Register And Decoder

بعد أن يتم إحضار الأمر من الذاكرة إلى شريحة المعالج لابد وأن يسجل أو يوضع فى أحد الأماكن فى انتظار تنفيذه ، هذا المكان هو مسجل الأوامر . أى أن مسجل الأوامر يحتوى شفرة الأمر الذى يتم تنفيذه الآن . لاحظ أن عدد بتات مسجل الأوامر عادة يساوى عدد بتات البايت فى الذاكرة التى تساوى بدورها عدد بتات مسار البيانات خاصة فى هذا الجيل من المعالجات الذى نحن بصدده الآن ، كما أن عدد الأوامر التى يمكن للمعالج أن ينفذها سيتوقف على عدد البتات فى مسجل الأوامر فمثلا إذا كان عدد بتات مسجل الأوامر هو 8 بت فإن ذلك يعنى أن هذا المعالج يستطيع التعامل مع 256= (2^8) أمر على الأكثر.

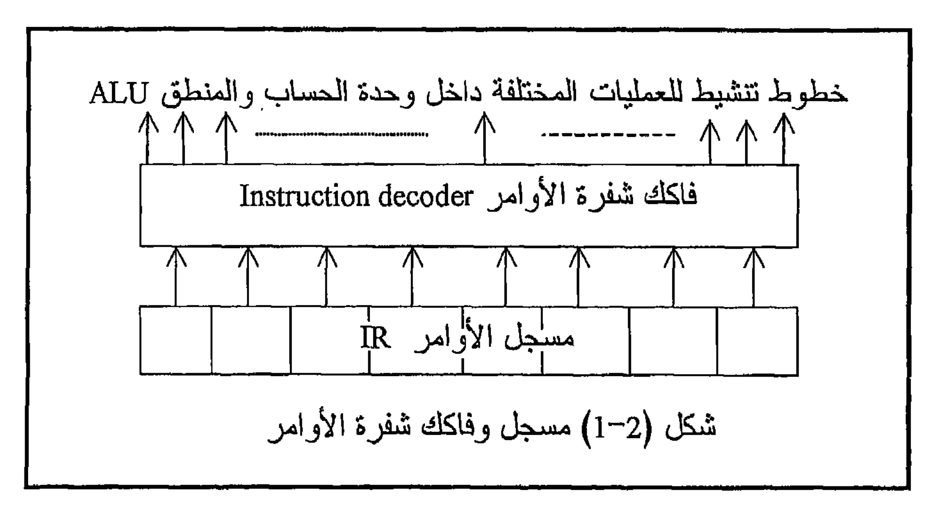
أول خطوات تنفيذ أى أمر تبدأ من فاكك شفرة الأوامر الذى يتصل دخله بخرج مسجل الأوامر كما فى شكل (2-1) بحيث أنه على حسب شفرة الأمر الموجودة فى مسجل الأوامر فإن عملية واحدة فقط سيتم تنفيذها على حسب الشفرة الموجودة على دخل فاكك الشفرة ويتم ذلك بالطبع بمساعدة وحدة التحكم ووحدة الحساب والمنطق .

2-4-4 مسجل الحالة Status Register, SR

أحيانا يطلق على هذا المسجل اسم مسجل الأعلام Flag Register, FR . يعتبر هذا المسجل نشرة إخبارية تعكس حالة نتيجة آخر عملية حسابية أو منطقية قام المعالج بتنفيذها ، فمن هذا المسجل نستطيع أن نعرف مثلا إذا كانت هذه النتيجة سالبة أم موجبة أم تساوى صفرا وغير ذلك من الأخبار المفيدة . هذا المسجل يحتوى على عدد من البتات وكل واحدة منها تعتبر علما flag يعكس أو يدل على حالة معينة من العملية الحسابية أو المنطقية التى تم تنفيذها ، من هذه الأعلام ما يلى :

2-4-4-1 علم الصفر Zero flag, ZF هذه البت تكون واحدا إذا كانت نتيجة آخر عملية حسابية أو منطقية تساوى صفرا وتكون هذه البت صفرا إذا كانت النتيجة مختلفة عن الصفر سواء موجبة أو سالبة .

2-4-4-2 علم الإشارة Sign flag, SF هذه البت تكون واحدا إذا كانت نتيجة آخر عملية حسابية أو منطقية نفذها المعالج سالبة ، أما إذا كانت هذه النتيجة موجبة فإن هذا العلم يكون صفرا ، لذلك فإنه أحيانا يسمى بعلم السالبية Negative Flag, NF . لاحظ أن آخر بت فى النتيجة تعكس إشارتها فإذا كانت آخر بت تساوى صفرا فإن ذلك يعنى أن النتيجة موجبة أما إذا كانت هذه البت واحدا فإن ذلك يعنى أن النتيجة سالبة لذلك فإنه دائما تكون محتويات علم الإشارة تساوى محتويات آخر بت فى النتيجة .

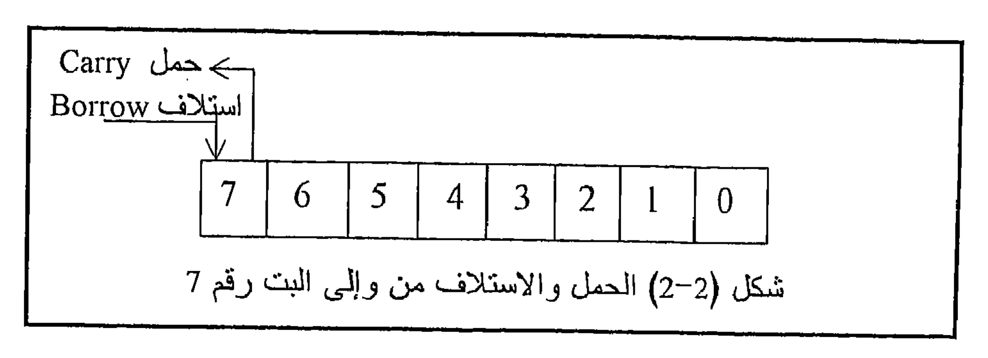


شكل (2-1) مسجل وفاكك شفرة الأوامر

2-4-4-3 علم الحمل Carry flag , CF هذا العلم يكون واحدا إذا حصل حمل carry من آخر بت فى أى عملية جمع أو حصل استلاف Borrow لآخر بت

في أي عملية طرح ويكون صفرا إذا لم يكن هناك حمل أو استلاف في آخر عملية حسابية . شكل (2-2) يبين كل من عمليات الحمل والاستلاف من وإلى البت رقم 7 .

2-4-4-4 علم الباريتى Parity flag, PF هذا العلم يكون واحدا إذا كانت آخر عملية حسابية أو منطقية قام بها المعالج تحتوى على عدد زوجي من الوحايد أما إذا كانت هذه النتيجة تحتوى على عدد فردى من الوحايد فإن هذا العلم يكون صفرا .



شكل (2-2) الحمل والاستلاف من وإلى البت رقم 7

2-4-4-5 علم الحمل النصفي أو البيني Half carry flag , HC هذا العلم يكون واحدا إذا كان هناك حمل من الخانة أو البت الثالثة إلى البت الرابعة نتيجة أي عملية جمع أو هناك استلاف من البت الرابعة إلى البت الثالثة نتيجة أي عملية طرح ، ويكون صفرا فيما عدا ذلك أى إذا لم يحدث استلاف أو حمل من أو إلى البت الرابعة ، لاحظ أننا هنا نبدأ عملية عد البتات بالرقم صفر ، أى أن أول بت هى البت رقم صفر . شكل (2-3) يبين كيفية تأثر علم الحمل النصفى .
التطبيق على جميع هذه الأعلام واستخدامها سيأتى عند الشرح التفصيلى لأوامر المعالج ، مع العلم أن عدد الأعلام سيختلف من معالج لآخر كما سنرى عند دراستنا للتركيب التفصيلى لكل بروسيسور سندرسه فى هذا الكتاب ولكن دعنا الآن ننظر للمثال التالى كتطبيق سريع على هذه الأعلام .

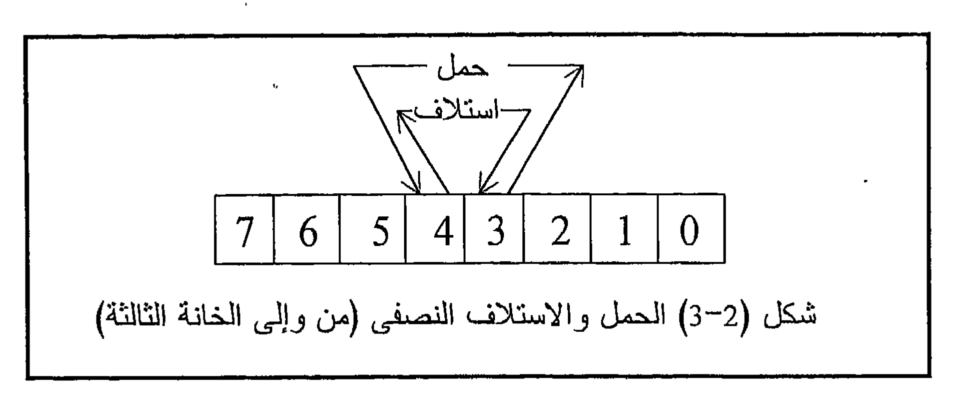
مثال 2. 1

اكتب محتويات الأعلام السابقة بعد إجراء عملية جمع الرقمين 77H و A5H .
لاحظ أن الرقمين مكتوبين فى الصورة الستعشرية hexadecimal .
الجمع الثنائى للرقمين السابقين سيتم كما يلى :

الرقم الأول	0111 0111
الرقم الثانى	1010 0101
النتيجة	0001 1100 حمل 1

نلاحظ الآتى من النتيجة السابقة :

1. النتيجة لا تساوى الصفر ، إذن فعلم الصفر يساوى صفر ZF=0 .
2. آخر بت فى النتيجة صفر فالنتيجة موجبة وعلم الإشارة يساوى صفر SF=0 .
3. هناك حمل من البت السابعة (الأخيرة) فعلم الحمل يساوى واحد CF=1 .
4. النتيجة تحتوى ثلاثة وحايد (عدد فردى) فعلم الباريتى يساوى صفر PF=0 .
5. ليس هناك حمل من الخانة الثالثة للرابعة فعلم الحمل النصفى يساوى صفر HCF=0 .



شكل (2-3) الحمل والاستلاف النصفى (من وإلى الخانة الثالثة)

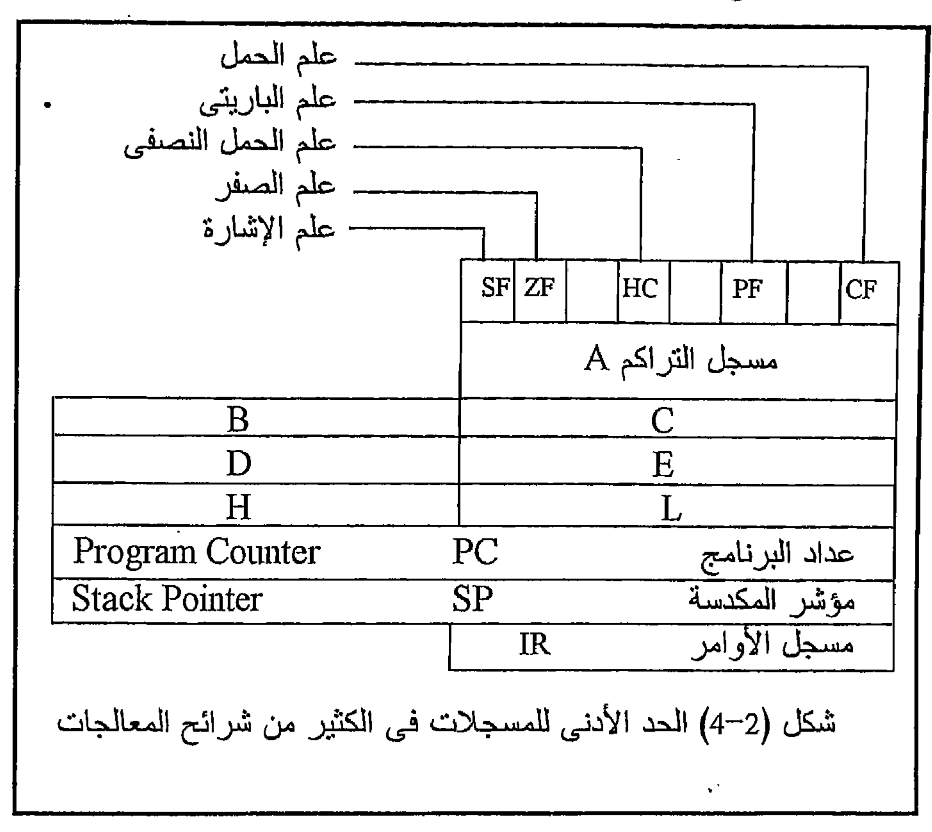
2-4-5 مسجل مؤشر المكدسة Stack Pointer register, SP

سيأتى إن شاء الله شرحا تفصيليا للمكدسة stack فيما بعد فى معرض الكلام عن البرامج الفرعية وبرامج المقاطعة ، ولكن الآن بإمكانك أن تعرف أن المكدسة هى جزء من الذاكرة يتم فيه تخزين بعض العناوين أو البيانات المهمة والتى لابد من الحاجة إليها واسترجاعها مرة ثانية وبنفس الترتيب الذى تم تخزينها به . مسجل مؤشر المكدسة يحتوى عنوان آخر مكان تم التسجيل فيه فى هذا الجزء من الذاكرة ، لذلك فإنه طالما أن هذا المسجل سيحتوى على عنوان فلابد أن يكون 16 بت . لاحظ أن المبرمج عادة تكون لديه الحرية فى اختيار الجزء من الذاكرة الذى سيستعمل كمكدسة .

2-4-6 المسجلات عامة الأغراض General Purpose Registers

فى الكثير من الأحوال عندما نجمع أكثر من رقم ، نحتاج لحفظ نتيجة معينة لحين استخدامها فى عملية أخرى لاحقة ، ولذلك فإنه بدلا من إرسال هذه النتيجة إلى الذاكرة ثم استدعائها ثانية مما يأخذ الكثير من الوقت فقد تم تجهيز المعالج ببعض المسجلات التى تستخدم لتخزين مثل هذه النتائج المرحلية لحين الحاجة إليها . عدد البتات فى هذه المسجلات يكون عادة مساويا لعدد بتات مسار البيانات . عدد هذه المسجلات يختلف من معالج لآخر ومن شركة لأخرى . ولقد

تم التعارف على تسمية هذه المسجلات بالمسجلات B و C و D و E و H و L كما سمى المركم من قبل بالمسجل A . هذه التسمية كما سنرى هى التسمية التى ستستخدم مع لغة الأسمبلى (التجميع) assembly language . شكل (2-4) يبين جميع المسجلات التى تكلمنا عنها حتى الآن والتى تمثل كما ذكرنا الحد الأدنى لمحتويات أى معالج من المسجلات .



شكل (2-4) الحد الأدنى للمسجلات فى الكثير من شرائح المعالجات

هناك بعض الأوامر التى تتعامل مع هذه المسجلات كأزواج يتكون كل زوج منها من 16 بت بدلا من التعامل معها كمسجلات يحتوى الواحد فيها على 8 بتات فقط. فى هذه الحالة يكون كل مسجل له مسجل آخر يمكن ازدواجه معه ولا يمكن ازدواجه مع أى مسجل آخر ، فمثلا المسجل B لا يزدوج إلا مع المسجل C فقط وكذلك المسجل D لا يزدوج إلا مع المسجل E والمسجل H لا يزدوج إلا مع المسجل L . لاحظ أنه فى حالة ازدواج المسجل B والمسجل C فإن المسجل C يحتوى أو يمثل البايت ذات القيمة الصغرى low significant byte من المعلومة المكونة من 16 بت والمسجل B يحتوى البايت ذات القيمة العظمى high significant byte من هذه المعلومة . بنفس الطريقة فى حالة الأزواج DE و HL

فإن المسجلات E و L تحتوى البايت ذات القيمة الصغرى والمسجلات D و H تحتوى البايت ذات القيمة العظمى . فمثلا إذا أردنا أن نسجل المعلومة 4CF6H المكونة من 16 بت فى زوج المسجلات HL فإن البايت F6 وهى البايت ذات القيمة الصغرى لابد أن توضع فى المسجل L وأما البايت 4C ذات القيمة العظمى فتوضع فى المسجل H . فى شكل (2-4) ستلاحظ أن هذه المسجلات موضوعة بنفس طريقة وكيفية ازدواجها .
إن التعامل مع هذه المسجلات من خلال المعالج يتم عن طريق شفرة أو كود code تم إعطاؤه لكل واحد من المسجلات العامة ولكل زوج منها بحيث يعرف كل مسجل فى لغة الماكينة كما سنرى فيما بعد بهذه الشفرة أو هذا الكود . إن هذه الشفرة كما هو موضح فى جدول 2-1 مكونة من وحايد وأصفار فقط وهذا يتناسب مع متطلبات لغة الماكينة . تذكر أيضا أن المكونات التى رأيناها إلى الآن ما هى إلا أقل ما يمكن أن يحتويه أى بروسيسور وإن اختلف عددها من بروسيسور لآخر كما سنرى .

الشفرة Code	المسجل Register
111	A
000	B
001	C
010	D
011	E
100	H
101	L
110	M
Register pairs	أزواج المسجلات
00	BC
01	DE
10	HL
11	SP

جدول 2-1 المسجلات وأزواج المسجلات وشفراتها الثنائية

2-5 نظرة خارجية على شرائح المعالج

إن مجموعة شرائح المعالجات ذى 8 بتات التى ندرسها فى هذا الكتاب كلها لها عدد 40 طرفا تخرج منها ، وكذلك المعالج 8086 ذو 16 بت . ابتداء من المعالج 80186 بدأ عدد أطراف هذه المعالجات فى الزيادة حيث أصبح 68 طرفا فى

المعالج 80186 ، وظل فى الزيادة إلى أن وصل إلى 296 طرفا فى حالة المعالج بنتيم برو Pentium Pro وهو آخر المعالجات التى سندرسها فى هذا الكتاب . فما هى وظيفة كل طرف من هذه الأطراف ، ولماذا كل هذا العدد من الأطراف ؟ إننا هنا سنحاول إلقاء نظرة سريعة على وظائف الأطراف الأساسية فقط وسوف نرجئ الحديث التفصيلى عنها وشكل الإشارات على كل طرف وكيفية ربط هذه الأطراف بالعالم المحيط بشريحة المعالج إلى فصول خاصة بذلك . هذه الأطراف يمكن تقسيمها إلى المجموعات التالية :

2-5-1 مسار العناوين Address bus

أى مكان يريد المعالج أن يتعامل معه سواء كان ذاكرة أو غيرها لابد وأن يحدد المعالج عنوانا لهذا المكان . هذا العنوان يتم وضعه فى صورة شفرات كهربية من الوحايد والأصفار بواسطة المعالج على عدد من هذه الأطراف الخارجة من المعالج تسمى مسار العناوين . لذلك فإنه على حسب عدد هذه الأطراف المخصصة لحمل شفرة العناوين يتحدد عدد الأماكن التى يمكن للبروسيسور أن يتعامل معها ، ودائما يكون عدد هذه الأماكن يساوى 2 مرفوعة لأس عدد هذه الخطوط أو الأطراف . فى جميع الشرائح 8 بت والتى نحن بصدد الكلام عنها يكون عدد أطراف مسار العناوين يساوى 16طرفا لذلك فإن مقدار الذاكرة التى يتعامل معها مثل هذا البروسيسور يساوى 2^{16}= 65536 بايت =64كيلوبايت باعتبار أن كل واحد كيلو بايت يساوى 1024 بايت . لاحظ أن الإشارة الموجودة على مسار العناوين تكون دائما خارجة من البروسيسور إلى الأجهزة الخارجية وليس العكس لأن البروسيسور هو فقط الذى يحدد العنوان الذى يريد التعامل معه . جدول 2-2 يبين علاقة بين عدد خطوط مسار العناوين وكمية الذاكرة التى يمكن التعامل معها فى كل حالة .

2-5-2 مسار البيانات Data bus

بمجرد أن يحدد المعالج المكان الذى يريد التعامل معه عن طريق العنوان الذى وضعه على مسار العناوين يقوم المعالج بإخراج أو استقبال المعلومة نفسها على أو من مسار آخر وهو مسار البيانات . هذا المسار أيضا عبارة عن عدد من الخطوط تصل بين المعالج والأجهزة المحيطة حيث تسير عليها البيانات المطلوب تداولها بين المعالج والأجهزة خارجه . إن عدد البتات التى تعرف بها أى شريحة معالج يكون على حسب عدد بتات أو أطراف مسار البيانات ، فالشرائح 8 بت سميت كذلك لأن لها مسار بيانات مقداره 8 بت والشرائح 16 بتا سميت كذلك لأن لها مسار بيانات 16 بتا وهكذا . فى عالم الحاسبات توضع أى معلومة دائما فى صورة عدد من الخانات أو البتات وكل بت أو خانة من هذه

الخانات يوضع بها واحد أو صفر حيث يمثل الواحد بجهد معين ويمثل الصفـــــر بجهد آخر . التركيبة المكونة من هذه الوحايد والأصفار هى ما يسمى بالشـــفرة الثنائية للمعلومة . إن ثمانية من هذه البتات أو الخانات تسمى بايت واثنين بـــايت تسمى كلمة أو Word . كمثال على هذه الشفرات الرقم 85H الذى شفرته هـــى 10000101 ، ولزيادة المعلومات عن نظم العد والتشفير المتبعة فـــى الحاسـب يمكن الرجوع إلى أى كتاب عن الإلكترونيات الرقمية . عندما يتعامل المعالج مع الذاكرة فإن وحدة التعامل بينهما تتوقف على عدد خطوط مسار البيانات لأن كـل بت من بتات المعلومة تنقل على خط منفصل . فى حالة الشرائح 8 بت فـإن أى معلومة تنقل من أو إلى المعالج لابد وأن تكون مكونة من ثمانيـــة بتـات ، إذا كانت هذه المعلومة مكونة من عدد من البتات أكبر من ثمانية فإنها تنقل علـــى أكثر من مرة وعلى حسب عدد بتاتها . فى حالة الشرائح 16 بتا تكـــون وحـدة التعامل فى نقل المعلومات هى 16 بتا ، لذلك فإنه مـــن البديـــهى أن نتوقـع أن الشرائح 16 بتا تكون أسرع من الشرائح 8 بت لهذا السبب أساسا وأسباب أخـرى سنعرفها فيما بعد ، فما بالك الآن بالشرائح 32 بتا والشرائح 64 بتا . لاحـظ أن زيادة عدد بتات مسار البيانات لن ينعكس فقط على سرعة التعامل مع الذاكـــرة ولكنه ينعكس أيضا على سرعة تنفيذ العمليات الحسابية . كلمة أخيرة عن مسـار البيانات وهى أن الإشارة عليه يمكن أن تكون خارجة من المعالج إلى الأجـهزة المحيطة أو داخلة إلى المعالج من الأجهزة المحيطة .

3-5-2 خطوط التحكم Control lines

هذه الخطوط يختلف عددها من معالج لآخر وعن طريق هذه الخطـــوط يخـبر المعالج أى جهاز من الأجهزة المحيطة (الذاكرة مثلا) الذى تم تحديد عنوانه على مسار العناوين عن الغرض من هذا التعامل ، فقد يكون الغرض من التعامل مـع الذاكرة مثلا هو القراءة منها ، أى استقبال معلومة منها ، فى هذه الحالـــة فـإن البروسيسور يرسل إشارة إلى الذاكرة على خط التحكم $\overline{MEMR}$,Memory Read تعرف منها الذاكرة أن الغرض من التعامل هو القراءة فتقوم بإرسال المعلومـــة المطلوبة على مسار البيانات فيتلقاها المعالج . أما إذا كان الغرض من التعـــامل هو الكتابة أو إرسال معلومة إلى الذاكرة فإن المعالج يقوم بوضع إشـــارة علـــى الخط $\overline{MEMW}$,Memory Write تفهم منها الذاكرة الغرض من التعامل فتتلقـــى المعلومة من على مسار البيانات . هناك خطان للتحكم بنفس الطريقة للتعامل مع بوابات الإخراج والإدخال . هناك أيضا خطوط المقاطعة Interrupt التى بها تتم مقاطعة أى برنامج يجرى تنفيذه وخطوط المسك HOLD التى بها يتـم فصـل البروسيسور عن المسارات لأغراض معينة .

عدد خطوط مسار العناوين	كمية الذاكرة التى يمكن التعامل معها
1	2 بايت
2	4
3	8
4	16
5	32
6	64
7	128
8	256
9	512
10	1024 واحد كيلوبايت (اكب)
11	2 كب
12	4 كب
13	8 كب
14	16 كب
15	32 كب
16	64 كب
17	128 كب
18	256 كب
19	512 كب
20	1024 كب (1 ميجابايت ، 1مب)
21	2 مب
22	4 مب
23	8 مب
24	16 مب
25	32 مب
26	64 مب
27	128 مب
28	256 مب
29	512 مب
30	1024 مب (1 جيجابايت , 1 جب)
31	2 جب
32	4جب
33	8 جب
34	16 جب
35	32 جب
36	64 جب

جدول 2-3 عدد خطوط مسار العناوين وكمية الذاكرة

هذه الأطراف وغيرها سيأتى الكلام بالتفصيل عنها فيما بعد نظرا لأن عددها وشكل الإشارة عليها يختلف من معالج لآخر . من أهم الأطراف التى يجب أن

نأخذ فكرة عنها هى طرف التزامن CLOCK وعلى هذا الطرف يتم إدخال نبضات كهربية بمواصفات معينة وتردد معين يحدد على حسب نوع شريحة المعالج . هذه النبضات CLOCK هى ساعة التوقيت الخاصة بالمعالج حيث يحدد زمن تنفيذ أى عملية يقوم بها المعالج بعدد معين من هذه النبضات يجب ألا تتعداه ، ولذلك فإن تردد هذه النبضات يعتبر خاصية من الخواص التى يعرف بها المعالج حيث بها أساسا تحدد سرعة المعالج . فى حالة المعالجات 8 بت يكون تردد التزامن CLOCK اثنين ونصف ميجاهرتز تقريبا قد تزيد أو تقل من معالج لآخر . إن ذلك يعنى أن زمن النبضة الواحدة حوالى نصف ميكروثانية تقريبا ، فإذا علمنا أن عملية جمع مسجلين مثلا تتم بعد 7 من هذه النبضات فإن ذلك يعنى أن عملية جمع المسجلين ستتم فى زمن مقداره ثلاثة ونصف ميكروثانية ! ، فما بالك بالمعالجات التى تبلغ نبضات الساعة لها الآن 400 أو 500 ميجاهرتز . جدول 2-3 يبين عدد خطوط مسار العناوين ومسار البيانات فى بعض المعالجات ، وكذلك سنة ظهور كل واحد منها . هذا الجدول يبين أيضا كمية الذاكرة التى يمكن لكل معالج من هذه المعالجات أن يتعامل معها . يبين الجدول أيضا تردد نبضات الساعة لكل معالج كمقياس لسرعة تنفيذ الأوامر . حاول دراسة هذا الجدول لتتبين التطور السريع فى بناء المعالجات .

رقم المعالج	سنة الظهور	عرض المسجلات	مسار العناوين	مسار البيانات	المدى العنوانى	تردد نبضات الساعة
8080	1974	8 بت	16 بت	8 بت	64 ك بايت	2 م هرتز
8085	1976	8 بت	16 بت	8 بت	64 ك بايت	2 م هرتز
Z80	1977	8 بت	16 بت	8 بت	64 ك بايت	2-4 م هرتز
8086	1978	16 بت	20 بت	16 بت	1 م بايت	6-16 م هرتز
80186	1980	16 بت	20 بت	16 بت	1 م بايت	6-16م هرتز
80286	1982	16 بت	24 بت	16 بت	16 م بايت	12-20م هرتز
80386	1985	32 بت	24 بت	16 بت	16 م بايت	16-40م هرتز
80486	1989	32 بت	32 بت	32 بت	4 ج بايت	25-66م هرتز
Pentium	1993	32 بت	32 بت	64 بت	4 ج بايت	60-200م هرتز
Pentium Pro	1995	32 بت	36 بت	64 بت	64 ج بايت	150-200 م هرتز

جدول 2-3 معلومات عامة عن المعالجات التى سيتناولها هذا الكتاب

6-2 شرائح المعالجات ذات 8 بت
8 bit microprocessors

سندقق النظر فى هذا الجزء على تركيب شريحتين من شرائح الجيل الثانى من المعالجات وهى الشرائح Intel8085 و Z80 ولقد اختيرت هذه الشرائح بالذات لأنها هى الأكثر استخداما وكانت وما زالت الأسهل فى التعلم والأبسط فى التركيب والأنسب لتقديم فكرة المعالج وكيفية عمله وبرمجته للمتعلمين الجدد فى هذا المجال .

2-6-1 الشريحة Intel8085

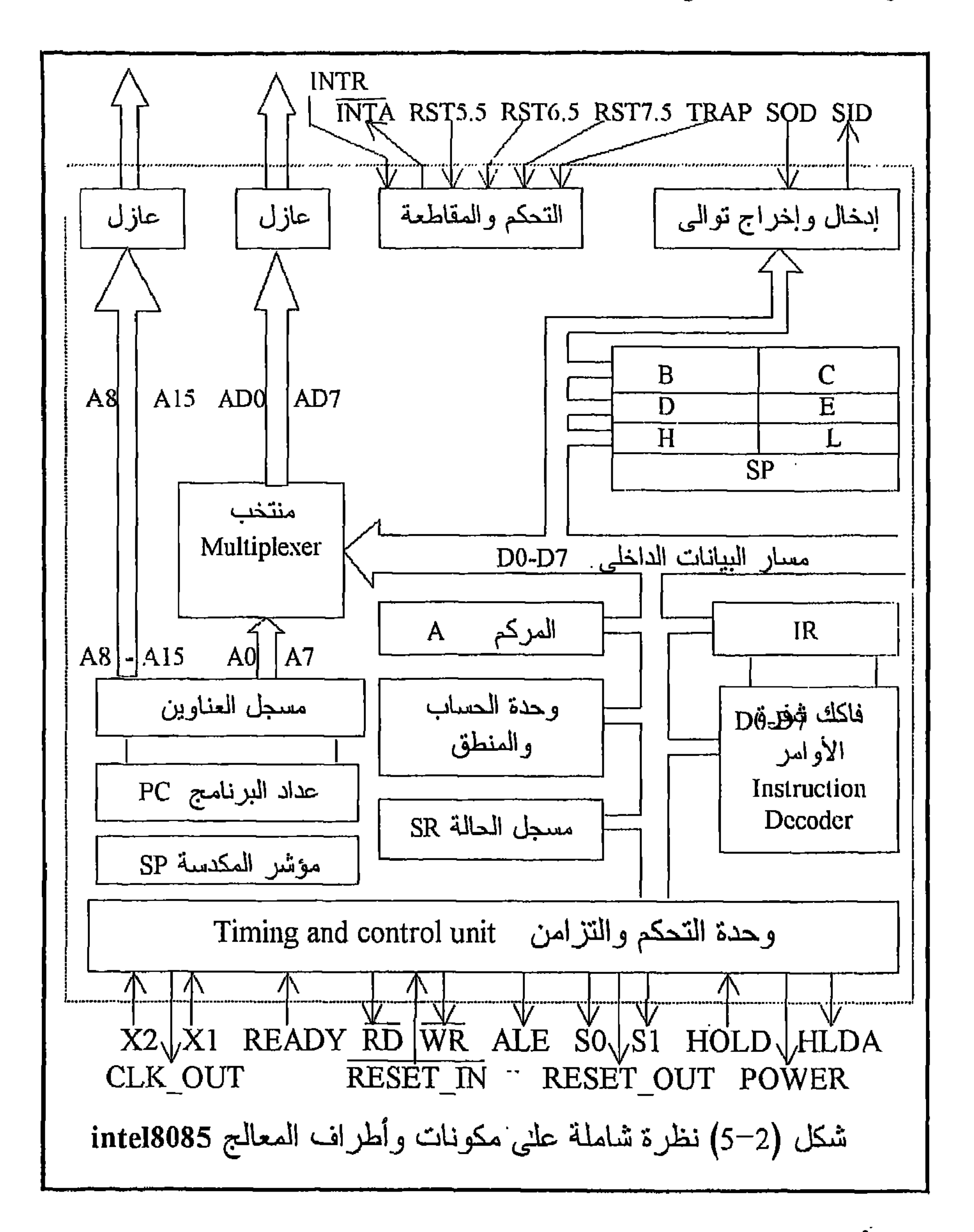
شكل (2-5) يبين المحتويات التفصيلية لشريحة المعالج 8085 ومن هذا الشكل يمكننا ملاحظة الآتى :

1. نلاحظ وجود الحد الأدنى من المسجلات والعدادات الذى ذكرناه من قبل وهو مسجل تراكم واحد A وعداد برنامج PC ومسجل ومشفر للأوامر IR ومسجل مكدسة SP ومسجل حالة SR بالإضافة لوحدة الحساب والمنطق وستة من المسجلات العامة .
2. لاحظ وجود مسجل للعناوين Address register وهذا المسجل 16 بت يحتوى عنوان أى مكان فى الذاكرة يراد التعامل معه . النصف العلوى من أى عنوان A15-A8 يخرج من مسجل العناوين إلى خارج الشريحة مباشرة من خلال عازل Buffer ، وأما النصف الأول A7-A0 فإنه يدخل أولا على مازج Multiplexer يقوم بدمج إشارة هذه الخطوط فى تتابع زمنى محدد مع الإشارة القادمة من مسار البيانات وإرسال الإشارتين على نفس الخطوط حيث تخرج إلى خارج الشريحة من خلال عازل أيضا . إن عملية المزج هذه التى يقوم بها المازج يقصد بها تقليل عدد أرجل الشريحة وأما كيفية فصل الإشارتين ثانية فسوف يتم الحديث عنه بالتفصيل فى فصل قادم .
3. عدد الأطراف الخارجة من الشريحة 40 طرفا سيأتى الحديث عن كل طرف وشكل الإشارة الموجودة عليه فى فصل قادم أيضا إن شاء الله . الخط المنقط فى شكل (2-5) عبارة عن حدود للشريحة يبين الأطراف الخارجة منها والتى من خلالها يتم الاتصال بين خارج الشريحة وداخلها . اتجاه السهم على هذه الخطوط يبين أيضا اتجاه الإشارة على كل منها إذا كانت داخلة للمعالج أم خارجة منه .

2-6-2 المعالج Z80

كما نلاحظ من شكل (2-6) فإن أساس تركيب الشريحة Z80 هو نفسه أساس تركيب الشريحة Intel8085 من حيث وجود المسجلات الأساسية مثل عداد البرنامج ومسجل ومشفر الأوامر ومسجل التراكم وعدد من المسجلات العامة

ووحدة الحساب والمنطق ووحدة التحكم والتزامن ، لكن كما ذكرنا من قبل دائما تكون هناك بعض الاختلافات عن هذا الأساس وتتمثل هذه الزيادات في حالة الشريحة Z80 فيما يلى :

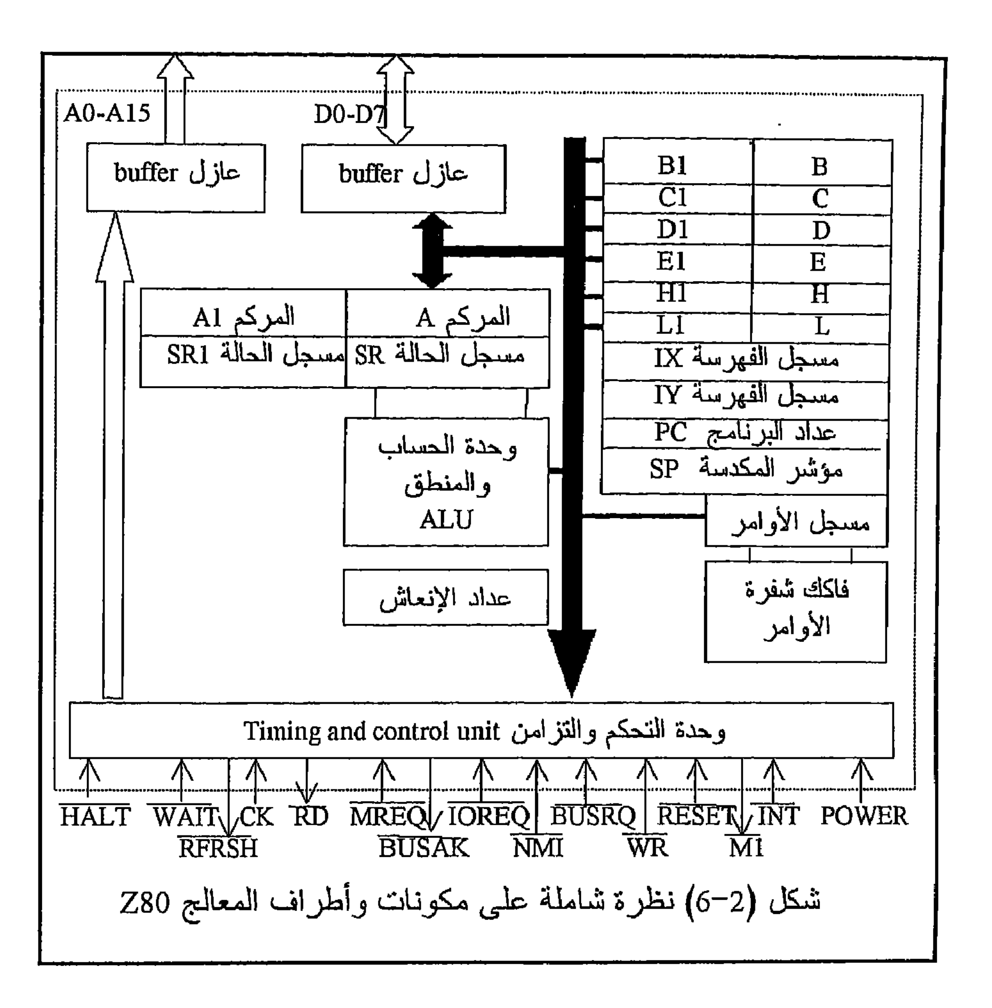


شكل (5-2) نظرة شاملة على مكونات وأطراف المعالج **intel8085**

1. معظم مسجلات الشريحة Z80 تم مضاعفتها فهناك مثلا مسجلين للتراكم A و A1 ومسجلين للحالة SR و SR1 وجميع المسجلات العامة تم مضاعفتها

أيضا كما هو موضح فى شكل (2-6) وذلك سيكون له ميزة عظيمة فى عمليـــة البرمجة كما سنرى .

2. تم زيادة المسجلين IX و IY وكل منهما 16 بت وهذان المسجلان يســتخدمان فى طرق مختلفة لعنونة الذاكرة كما سيتضح فيما بعد .



شكل (2-6) نظرة شاملة على مكونات وأطراف المعالج Z80

3. تم زيادة عداد إنعاش الذاكرة والذى يستخدم فـــى عمليـــة إنعـــاش الذاكـــرة الديناميكية لأنه كما نعلم فإن الذاكرة الديناميكية تحتاج دائما لعمليـــــة تجديـــد أو إنعاش أو إعادة تخزين لمحتوياتها بعد فترات زمنية محددة وإلا فإنها تفقدها بعـــد زمن مقداره بعض ميللثانية .

4. نظرة شاملة على الأطراف الخارجية للمعالج Z80 سنجد أن له عدد 40 طرفا لها تقريبا نفس الوظائف الخاصة بـــأطراف المعـــالج Intel8085 وإن اختلفـــت المسميات وسنعرف ذلك بالتفصيل عند دراسة الخواص والوظائف المختلفة لكـــل

طرف . الخط المنقط فى شكل (2-6) عبارة عن حدود للشريحة يبين الأطـــراف الخارجة منها والتى من خلالها يتم الاتصال بين خارج الشريحة وداخلها . اتجـاه السهم على هذه الخطوط يبين اتجاه الإشارة على كل منها إذا كانت داخلة للمعالج أم خارجة منه .

2-7 تـــمــــاريـــن

1. اذكر المهام الأساسية التى من المفروض أن يقوم بها أى معالج ؟
2. لماذا يعتبر مسجل التراكم من أهم المسجلات فى المعالج ؟
3. كم عدد بتات مسجل التراكم فى المعالجات التى ندرسها فى هـــذا الكتـــاب ؟ ولماذا هذا العدد بالذات ؟ وماذا يحدث لو نقص هذا العدد أو زاد ؟
4. هل تصنف مسجل التراكم من المسجلات عامة الأغراض general purpose registers أم المسجلات خاصة الأغراض dedicated registers ؟
5. ما هى وظيفة عداد البرنامج PC ؟ وكم عدد بتاته ؟ ولماذا يرتبط عدد بتاتـــه بعدد بتات مسار العناوين ؟
6. ماذا يحدث لو أن عدد بتات عداد البرنامج كان ثمانيـــة بـــدلا مـــن 16 فـــى المعالجات التى ندرسها ؟ وماذا يحدث لو أن هذا العدد كان 20 مثلا ؟
7. ما هى وظيفة مسجل الأوامر IR ؟ وكم عدد بتاته ؟ وهل يرتبط هذا العـــــدد بمسار البيانات data bus أم بمسار العناوين address bus ؟
8. ما هى وظيفة مسجل الحالة SR ؟ اذكر الأعلام الموجودة فى مسجل الحالـــة للمعالج الذى تهتم بدراسته فى هذا الكتاب ، ومتى يكون كل علم من هذه الأعلام واحدا ومتى يكون صفرا ؟
9. هل مسجل الحالة ومسجل الأوامر وعداد البرنامج تصنف على أنها مسجلات عامة الأغراض أم خاصة الأغراض ؟
10. على ضوء المهام المنوطة بالمعالج ، هل تشعر أن أيا من المسجلات الســابقة يعتبر زائدا ويمكن الاستغناء عنه ؟
11. ما هى محتويات كل علم من الأعلام بعد إجراء العمليات التالية :

10101111	10101111	01101110	11011101
11110001 XOR	11110001 AND	00101111−	10011001+

12. هل يحتوى المعالج الذى تهتم بدراسته على مسجلات عامة الأغراض غـــير مسجل التراكم ؟ أذكر هذه المسجلات ، وما فائدتها ؟ وهل يمكن الاستغناء عنها؟
13. وضح بالرسم تركيب المعالج الذى تهتم بدارسته ؟

14. الكلية التى ندرس بها فيها 200 عضوا هيئة تدريس ، مطلوب إعطاء شفرة ثنائية لكل واحد منهم ، وكم سيكون عدد بتات هذه الشفرة ؟ هذه الشفرة الثنائية ، هل يمكن التعارف على أنها بمثابة عنوان للشخص ؟
15. شفرة ثنائية مكونة من 5 بتات ، كم عدد العناوين التى يمكن تشفيرها بهذا العدد من البتات ؟
16. كم عدد الخطوط (البتات) فى مسار العناوين فى المعالج الذى تدرسه ؟
17. ما مقدار كمية الذاكرة التى يستطيع أن يتعامل معها هذا المعالج ؟
18. ما هو تأثير زيادة أو نقصان عدد الخطوط فى مسار العناوين لأى معالج ؟
19. لدينا 16 راكبا نريد نقلهم من مكان إلى مكان آخر باستخدام أتوبيس يسع 8 ركاب فقط ، كم عدد المشاوير التى سيقوم بها الأتوبيس ؟ لو استخدمنا أتوبيس يسع 16 راكبا ويسير بنفس سرعة الأول ، كم سيكون عدد المشاوير؟ وأى الوسيلتين أسرع ؟
20. لو شبهنا الأتوبيس بمسار البيانات للمعالج ، وسعة الأتوبيس بعدد البتات (الخطوط) فى هذا المسار ، أيهما سيكون أفضل من حيث السرعة فى نقل المعلومات ، المعالج ذو 8 بتات أم ذو 16 بتا ؟

الفصل الثالث

برمجة المعالج

Microprocessor Programming

3-1 مقدمة

بعد أن أخذنا فكرة عامة عن وظيفة ومهمة كل مسجل من مسـجلات شـريحة المعالج سنتعرض فى هذا الفصل وبعض الفصول القادمة إلى كيفية برمجة هـذه الشريحة وهذا هو الشق الأول من دراسة شرائح المعالجات كما ذكرنـا ، وأمـا الشق الآخر وهو مواجهة المعالج فإن ذلك سيكون فى فصول قادمـة أخـرى إن شاء الله . سنتعرض فى هذا الفصل لدراسة الأفكار العامة عـن لغـة التجميـع (الأسمبلى) دون أن نخص بالذكر أى شريحة معينة حيث سيعقب هـذا الفصـل فصل خاص بلغة الأسمبلى والأوامر الخاصة بكل واحدة من شـرائح المعـالج Intel8085 و Z80 .

3-2 لغات الحاسب Computer languages

لغات الحاسب يمكن تقسيمها إلى قسمين أساسيين :
القسم الأول : وهو اللغات التى تعتمـد علـى الماكينـة machine dependent languages وكل لغة من هذا النوع تكون مصممة لماكينة معينة وما يتوافق منـها مع ماكينة معينة ليس بالضرورة أن يتوافق مع الماكينات الأخرى . من أمثلة هذا النوع من اللغات ، لغة الماكينة machine language ولغة الأسـمبلى assembly language حيث أن كل معالج له مجموعة الأوامر الخاصة بـه التـى عـادة لا تتوافق مع المعالجات الأخرى . فأنت مثلا إذا كتبت برنامجـا بلغـة الأسـمبلى الخاصة بالشريحة MC6800 فإن هذا البرنامج لا يمكن أن ينفـذ مـع الشـريحة Intel8085 أو الشريحة Z80 .
القسم الثانى : هو اللغات التى لا تعتمد علـى الماكينـة machine independent language ومن أمثلة هذه اللغات جميع اللغات ذات المستوى العـالى high level languages مثل الفورتران والبسكال و PL/1 و PL/C وغير ذلك من هذه اللغات التى تعد بالمئات الآن .

3-3 ما هو الأمر؟

الأمر معناه الكود أو الشفرة الثنائية التى تعطى للمعالج والتى على أثرهـا يقـوم بعمل فعل معين . هذا الفعل قد يكون عملية جمع رقمين أو إحضار معلومة مـن الذاكرة أو غير ذلك من الأفعال التى يستطيع المعالج القيام بها . من أمثلة هـذه الشفرات الثنائية الشفرة 10000000 والتى معناها اجمع محتويات المسجل B مـع

مسجل التراكم A وضع النتيجة فى المسجل A . كما نرى فإن المعالج يتعـــرف فقط على الشفرات الثنائية ولا يعرف أى نوع آخر من الشـــفرات ســـواء كـــانت حرفية أو ثمانية أو ستعشرية . لقد سبق تعريف كل من الأمـــر والبرنـــامج فـــى الفصل الأول بصورة عامة ولكننا أعدنا تعريفهما فى هذا الفصـــل بشـــىء مـــن التفصيل .

3-4 ما هو البرنامج ؟

00111010
01100000
00000000
01000111
00111010
01100001
00000000
10000000
00110010
01100010
00000000

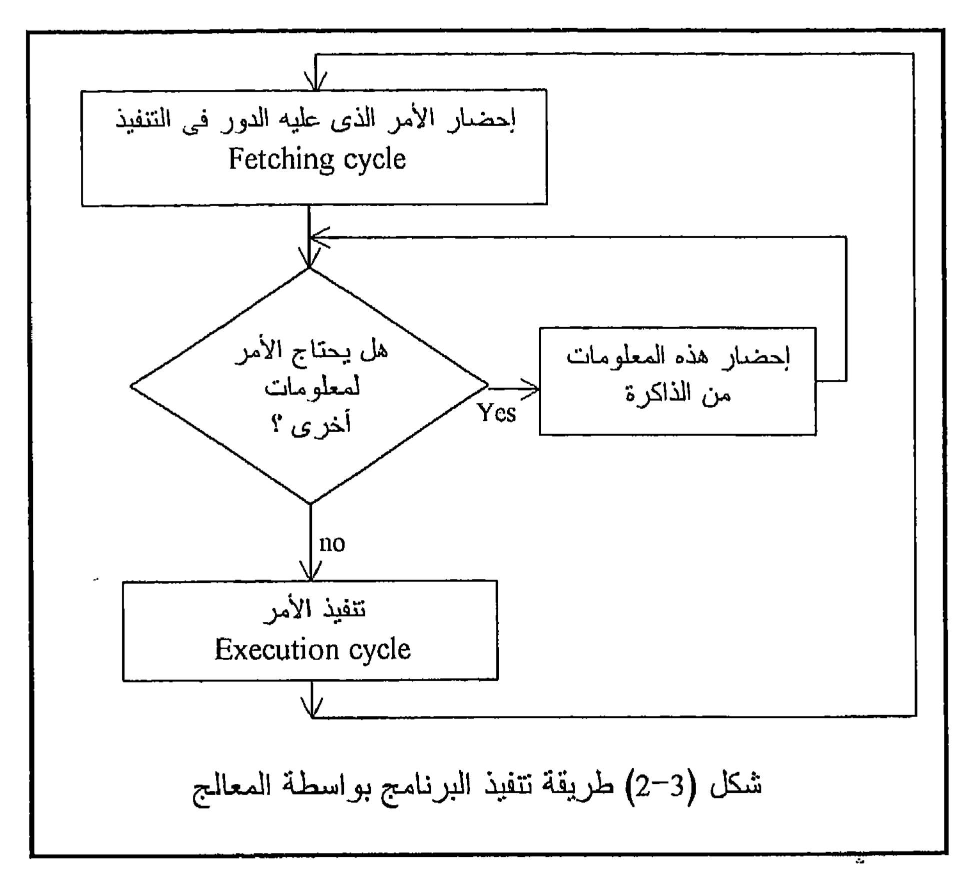
شكل (3-1) برنامج يجمع محتويات العنوان 60H مع العنوان 61H ويضع النتيجة فى العنوان 62H

البرنامج عبارة عن مجموعة من الأوامر التى ينتج عن تنفيذها مجتمعة هـــدف أو عمل معين كإدارة موتور مثلا أو التحكم فى متغير معين أو التعرف على معلومة معينة من بين الكثير من المعلومات . يجب أن يحتوى البرنامج أو يحدد مصـــدر أى معلومة يتم استخدامها بواسطته ، فمثلا لإجراء عملية جمع لرقمين لابـــد وأن يحدد البرنامج أين يوجد الرقمان وأين ستوضع النتيجة . يمكن النظر لأى برنامج على أنه مجموعة من الشفرات الثنائية المخزنة فى الذاكرة فى انتظـــار أن يقـــوم المعالج بتنفيذها . كمثال على ذلك أنظر إلى البرنامج الموجود فى شـــكل (3-1) دون أن تبذل أى مجهود فى محاولة فهمه الآن . إن هذا البرنامج يقـــوم بجمـــع محتويات عنوان الذاكرة رقم 60H مع محتويات العنوان رقم 61H ويضع النتيجـــة فى العنوان 62H ، إن الحرف H يعنـــى أن هـــذه الأرقـــام مكتوبـــة بالنظـــام الستعشرى. هذا البرنامج نقول أنه مكتوب بلغـــة الماكينـــة machine language وعادة يطلق عليه اسم برنامج الهدف object program .

3-5 كيف يقوم المعالج بتنفيذ البرنامج ؟

شكل (3-2) يبين الخطوات التى يقوم بها المعالج لكى يتم تنفيذ أى برنامج و هى كالتالى :

1. يقوم المعالج (وحدة التحكم بداخله) بقراءة الأمر الأول من الذاكرة وتخزينه فى مسجل الأوامر IR .



شكل (3-2) طريقة تنفيذ البرنامج بواسطة المعالج

2. يقوم المعالج بفك شفرة هذا الأمر أو بمعنى آخر يتم التعرف على هذا الأمر من بين قائمة أوامر المعالج/ وعلى ضوء هذا التعارف يقرر المعالج إذا كان هذا الأمر سيحتاج لمعلومات أخرى من الذاكرة لكى تتم عملية التنفيذ أم لا ؟ وإذا كان الأمر سيحتاج لمثل هذه المعلومات يقوم المعالج بإحضارها أيضا من الذاكرة. بذلك تنتهى المرحلة الأولى من مراحل تنفيذ الأمر الأول وهى مرحلة الإحضار fetching cycle .

3. بمجرد الانتهاء من مرحلة أو دورة الإحضار تبدأ مرحلة التنفيذ execution cycle حيث يقوم مشفر الأوامر مع وحدة التحكم بإرسال الإشارات المناسبة إلى وحدة الحساب والمنطق التى تقوم بتنفيذ هذا الأمر .
4. بعد الانتهاء من مرحلة تنفيذ الأمر الأول يرجع المعالج إلى الخطوة الأولى حيث يبدأ فى عملية إحضار الأمر الثانى ثم يتم تنفيذه ثم يبدأ فى عملية إحضار الأمر الثالث وتنفيذه وهكذا حتى ينتهى البرنامج .

3-6 طريقة كتابة البرنامج للمعالج

3-6-1 الشفرات الثنائية Binary codes

إن الناظر لأول وهلة فى البرنامج المكتوب بلغة الماكينة فى شكل (3-1) سيصاب بالذهول وسيقول : هل من المعقول أن أكتب كل هذا العدد من الوحايد والأصفار فى برنامج لا يتعدى الأربع خطوات ؟ بالطبع إن ذلك حق لأن كتابة البرامج بلغة الماكينة تصاحبها بعض العيوب والتى نوردها فيما يلى :

1. هذه البرامج تأخذ وقتا طويلا فى إدخالها للذاكرة لأننا نكتبها بت بُعد بت .
2. مثل هذه البرامج من الصعب فهمها أو متابعتها أو تصحيح أى خطأ فيها وذلك لأن الأعداد الثنائية عبارة عن نمازج من الوحايد والأصفار التى يصعب التفريق بينها خاصة بعد فترة عمل طويلة مع هذه الأرقام .
3. شكل هذا البرنامج لا يعطى أى دلالة على الغرض منه ، على العكس من برامج الباسيك أو حتى الأسمبلى كما سنرى بعد قليل فإنه بعد نظرة فاحصة على البرنامج تستطيع أن تخبر ما الغرض منه .
4. من السهل أن يقع المبرمج فى الكثير من الأخطاء أثناء كتابة هذه البرامج ومن الصعب عليه جدا استخراج هذه الأخطاء فيما بعد .

كمثال على ذلك سنكتب البرنامج السابق الموجود فى شكل (3-1) مرة أخرى فى شكل (3-3) وبجانبه صورة منه تحتوى على خطأ معين وحاول استخراج هذا الخطأ !! ما رأيك الآن لو كان البرنامج مكونا من عشرات أو حتى مئات من السطور هل تستطيع استخراج أخطائه كلها ؟ لاحظ أنك فى شكل (3-3) أمامك الصورة الصحيحة والصورة الخطأ وأنت فقط تقارن الاثنين ولكن عادة فى الوضع الحقيقى فإنه لن تكون أمامك الصورة الصحيحة للبرنامج ولكنك أخبرت بأن البرنامج به خطأ وعليك استخراجه ، بالطبع فإن هذه ستكون عملية شاقة .

3-6-2 الشفرات الستعشرية Hexadecimal codes

من الممكن تسهيل عملية كتابة البرامج بلغة الماكينة عن طريق استخدام نظام آخر غير النظام الثنائى وليكن مثلا النظام الستعشرى أو النظام الثمانى وسنعرض

هنا للنظام الستعشرى فقط على أساس أنه الأكثر شيوعا وأنه الأسهل فى عملية الكتابة لأن عدد خانات العدد بالنظام الستعشرى تكون عادة أقل منها فى النظام الثمانى .

00111010	00111010
01100000	01100000
00000000	00000000
01000111	01000111
01110010	00111010
01100001	01100001
00000000	00000000
10000000	10000000
00110010	00110010
00110010	00110010
00000000	00000000

شكل (3. 3) صعوبة استخراج الأخطاء فى ظل الكتابة بلغة الماكينة

شكل (3-4) يبين برنامج الجمع السابق وقد تمت كتابته هذه المرة بالنظام الستعشرى . من هذا الشكل نلاحظ أن عملية كتابة البرامج باستخدام النظام الستعشرى وكذلك عملية فحص البرنامج واستخراج الأخطاء منه ستكون أسهل بكثير من استخدام النظام الثنائى فى ذلك . المشكلة الآن هى أنه كما سبق وذكرنا أن المعالج لا يعرف سوى الإشارات المكتوبة بالنظام الثنائى فقط (وحايد وأصفار) فما هو الحل وقد كتبنا البرنامج بالنظام الستعشرى كما هو فى شكل (3-4) ؟ إن الحل لهذه المشكلة هو تحويل هذه الأوامر من الصورة الستعشرية إلى الصورة الثنائية قبل أن يتم إدخالها فى الذاكرة !! ولكن من سيقوم بعملية التحويل هذه ؟ بالطبع إذا قام بها المستخدم فقد رجعنا إلى المشكلة الأولى وذلك لصعوبة عملية التحويل . إن هذه المهمة ، مهمة التحويل من الصورة الستعشروية إلى الصورة الثنائية ، هى مهمة سهلة جدا لأن يقوم بها المعالج نفسه عن طريق كتابة برنامج بلغة الماكينة (فى النظام الثنائى) يتلقى الأوامر من المستخدم بالنظام الستعشرى ثم يقوم هو (البرنامج) بتحويلها إلى النظام الثنائى وتحميلها فى الذاكرة . إن هذا البرنامج يسمى "محمل النظام الستعشرى" hexadecimal loader .

لقد حل النظام الستعشرى مشكلة صعوبة كتابة البرامج واستخراج الأخطاء منها إلى حد ما ، ولكن بقيت المشكلة الأخرى وهى أننا ما زلنا نتعامل مع أرقام صماء

كشفرات للأوامر لا تحمل أى دلالة عن ماذا يفعل هذا الأمر أو ذاك . فمثلا الرقم 3A هو شفرة لأمر معين ولكننا لا نستطيع مثلا أن نميز ذلك الأمر فقد يكون هذا الرقم مثلا جزءا من عنوان كالأرقام 62 ,61 ,60 وغيرها ، وحتى إذا عرفنا أنه شفرة لأمر فلن نستطيع معرفة ماذا يفعل هذا الأمر إلا إذا رجعنا إلى كتالوج خاص بذلك .

3A
60
00
47
3A
61
00
80
32
62
00

شكل (3-4) نفس البرنامج الموجود فى شكل (3-3) ولكن مكتوب بالنظام الستعشرى

3-6-3 الشفرات الحرفية Mnemonics codes

إنها لفكرة عظيمة لو أننا فهمنا المقصود من كل أمر من الأوامر وأعطينا كل واحدا منها كودا أو شفرة مكونة من ثلاثة أو أربعة أحرف على الأكثر على أن تكون هذه الأحرف من الأحرف الأبجدية التى تدل تقريبا على ما يقوم به المعالج عند تنفيذ هذا الأمر . فمثلا أمر الجمع يكون ADD التى هى اختصارا لكلمة Addition يعنى جمع ، وأمر الطرح يكون SUB وجاءت من Subtraction بمعنى طرح ....وهكذا مع باقى الأوامر كما سنرى فيما بعد . إن هذه الاختصارات هى ما يسمى بلغة الأسمبلى Assembly language أو أحيانا تسمى Mnemonics codes بمعنى الشفرات التى من السهل تذكرها حيث كلمة mnemonics تعنى المساعد لعملية التذكر ، وهى كذلك فى الحقيقة ، إذ الآن بوضع الأوامر فى هذه الصورة الحرفية أصبح من السهل تذكرها بل ومن السهل أن تخبر ماذا يفعل الأمر بمجرد النظر إليه . ما أروعها لو أن العرب قد سبقوا فى هذا المجال وفرضوا الكلمات طرح وجمع بدلا من SUB و ADD .....!

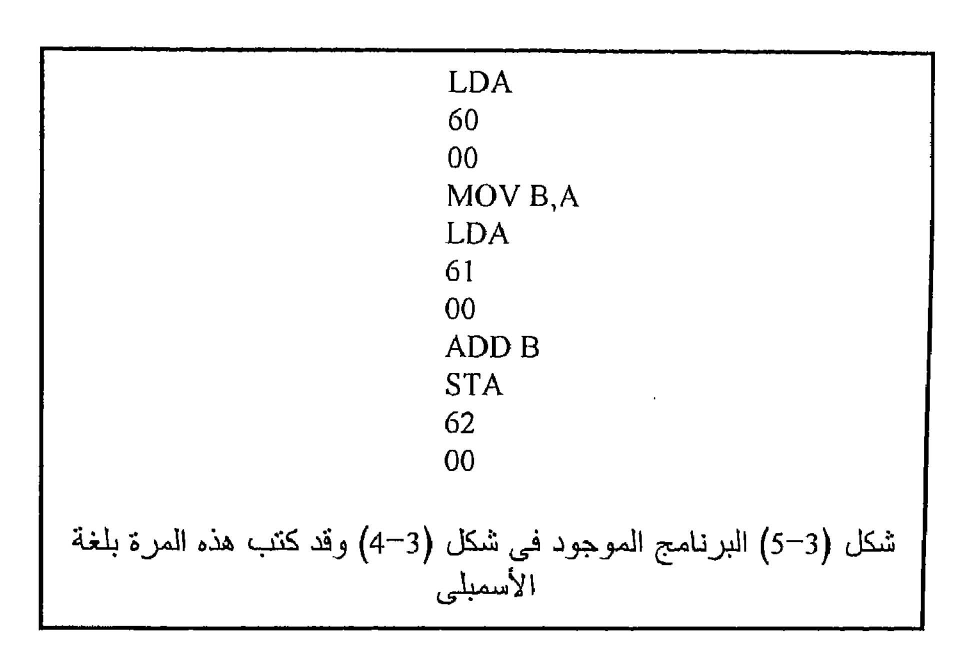
يقوم كل صانع لشريحة من شرائح المعالج بتزويدها بقائمة أو كتالوج يحتوى كل هذه الاختصارات الحرفية mnemonics ، ولذلك فإنك ستجد أن اختصارات كل

شركة منتجة تختلف عن اختصارات الشركات الأخرى وسوف نرى ذلك فى الفصول القادمة إن شاء الله . إن أى برنامج مكتوب بهذه الاختصارات يقال عنه أنه مكتوب بلغة الأسمبلى . شكل (3-5) يبين البرنامج الموجود فى شكل (3-4) والذى سبق كتابته بالشفرات الستعشرية وقد كتبت جميع أوامره هذه المرة بلغة الأسمبلى لأحد المعالجات ، انظر لهذا البرنامج وحاول توقع ماذا يفعل كل أمر من هذه الأوامر قبل أن ندرسها بالتفصيل .

نحن هنا أيضا أمام مشكلة ترجمة هذه الشفرات الحرفية mnemonics التى لا يستطيع المعالج التعرف عليها إلى شفرات ثنائية يعرفها المعالج ، وكما هى الحال مع الشفرات الستعشرية فإننا سنترك أمر هذه الترجمة ليقوم بها المعالج نفسه عن طريق برنامج مكتوب لهذا الغرض يقوم المعالج بتنفيذه فيحول هذه الشفرات الحرفية إلى الشفرات الثنائية المطلوبة . هذا البرنامج يطلق عليه الأسمبلر Assembler . على ذلك نستطيع القول أن الأسمبلر هو برنامج مكتوب بلغة الماكينة يقوم بتحويل البرنامج المكتوب بلغة الأسمبلى (الشفرات الحرفية) إلى برنامج مكتوب بلغة الماكينة . عادة يطلق على البرنامج المكتوب بلغة الأسمبلى "برنامج المصدر" source program والبرنامج المكتوب بلغة الماكينة "برنامج الهدف" object program . شكل (3-6) يبين رسما توضيحيا للدور الذى يقوم به الأسمبلر .

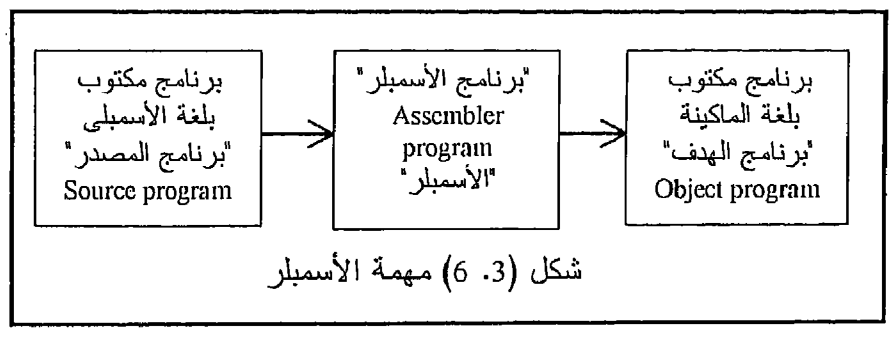
إن المقابل الذى يدفعه المستخدم نتيجة استخدامه للغة الأسمبلى يكون أولا فى كمية الذاكرة التى يشغلها برنامج الأسمبلر حيث أن هذا البرنامج لابد وأن يشغل كمية من الذاكرة الأساسية للميكروكومبيوتر وثانيا بعض التأخير الذى يحدث نتيجة الوقت الذى يأخذه الأسمبلر فى عملية الترجمة . وكما نعرف فإنه ليس هناك شيء كامل على الإطلاق ، لذلك فإننا نستطيع أن نلخص بعض عيوب لغة الأسمبلى فيما يلى :

1. مازالت هذه الاختصارات الحرفية غير كافية للدلالة على معنى الأوامر المختلفة حيث مازالت صيغ الأوامر بعيدة كل البعد عن اللغة العادية التى يستخدمها الإنسان وأيضا بالمقارنة بأوامر اللغات ذات المستوى العالى فإن لغة الأسمبلى تعتبر الأصعب فى التعلم .
2. لكى تستخدم هذه اللغة لابد من المعرفة الكاملة بمكونات المعالج ، المسجلات الموجودة بداخلة ، وطريقة المعالج فى التعامل مع الذاكرة وغير ذلك من الأمور الغير موجودة فى اللغات ذات المستوى العالى .
3. هذه اللغة كما ذكرنا من قبل تعتبر من اللغات التى تعتمد على الماكينة ، فأنت إذا كتبت برنامجا للمعالج Z80 فلن تستطيع استخدامه مع المعالج MC6800 مثلا.



```
LDA
60
00
MOV B,A
LDA
61
00
ADD B
STA
62
00
```

شكل (3–5) البرنامج الموجود فى شكل (3–4) وقد كتب هذه المرة بلغة الأسمبلى



شكل (3. 6) مهمة الأسمبلر

7–3 اللغات ذات المستوى العالى High level languages

لقد تم التغلب على الكثير من الصعوبات والعيوب المصاحبة للغة الأسمبلى باستخدام اللغات ذات المستوى العالى . إن كل أمر من أوامر أى لغة من هذه اللغات يدل أو يقوم بعملية مركبة على عكس لغة الأسمبلى فإن كل أمر فيها يقوم بعملية أولية . فمثلا برنامج الجمع السابق الذى يجمع رقمين موجودين فى الذاكرة والذى تمت كتابته فى أحد عشر سطرا باستخدام لغة الأسمبلى يمكن كتابته فى سطر واحد باستخدام لغة الباسيك كما يلى :

SUM = NUM1 + NUM2

لذلك فإن أى أمر من أوامر أى لغة من اللغات ذات المستوى العالى هو فى الحقيقة مجموعة من أوامر لغة الأسمبلى . إن ما يقوم به برنامج الأسمبلر فى حالة لغة الأسمبلى يقوم به برنامج آخر يسمى "برنامج المؤلف" أو الجامع أو المصنف compiler program فى حالة اللغات ذات المستوى العالى ، حيث يقوم هذا المؤلف بترجمة الأوامر المكتوبة باللغات ذات المستوى العالى إلى لغة الماكينة أو الشفرات الثنائية التى يقبلها المعالج . هذه اللغات ليست موضوع دراستنا فى هذا الكتاب لذلك سنكتفى بهذا القدر من الكلام عنها .

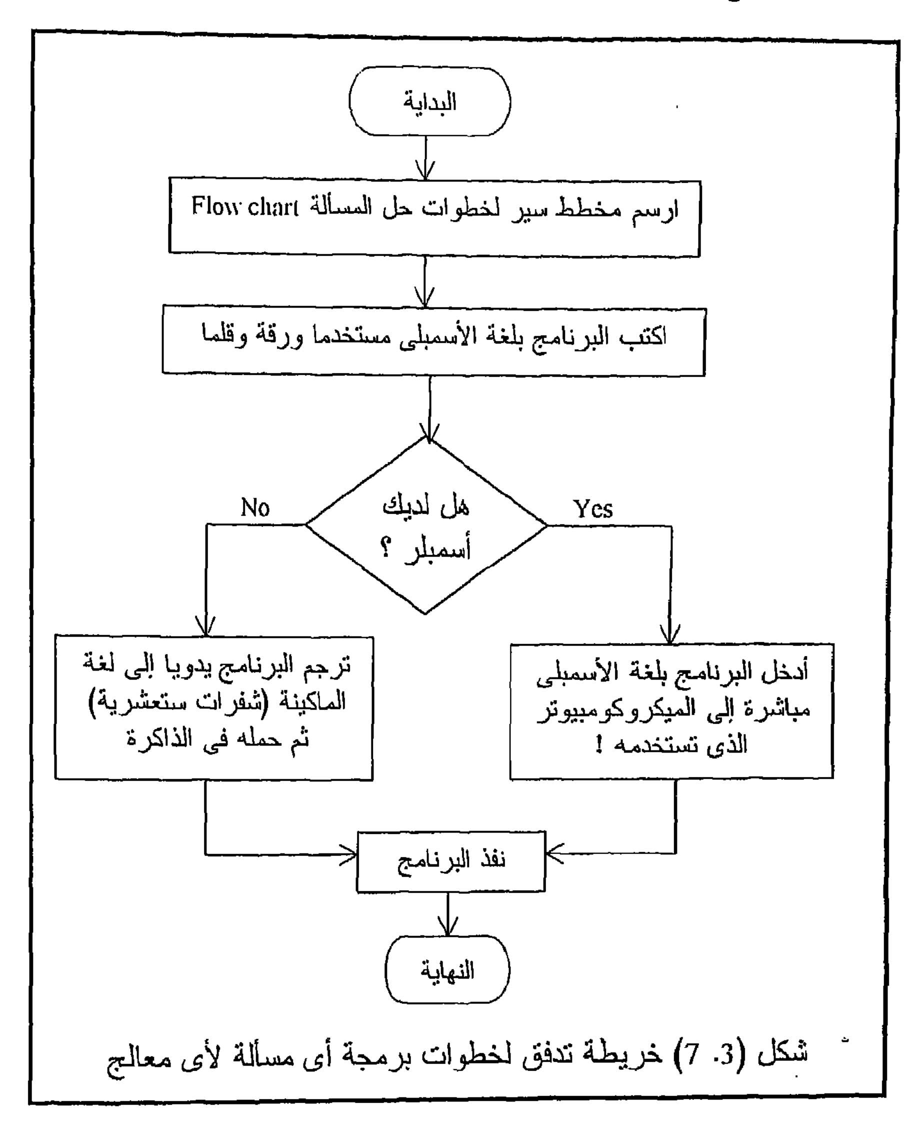
3-8 خطوات كتابة برنامج بلغة الأسمبلى

الآن وقد عرفنا الفرق بين لغة الماكينة ولغة الأسمبلى فبأى صورة سنكتب برامجنا ؟ الإجابة على ذلك ستتوقف على إمكانيات الميكروكومبيوتر الذى تتعامل معه سواء فى بيتك أو فى معملك . إذا كان لديك برنامج الأسمبلر فإنه بالطبع من الأفضل أن تكتب برامجك بلغة الأسمبلى على الحاسب بنفس الشكل ثم تطلب من الأسمبلر أن يقوم بعملية الترجمة وإدخال البرنامج إلى الذاكرة ، أما إذا لم يكن لديك أسمبلر فليس أمامك من خيار سوى الكتابة بلغة الماكينة أو الشفرات الستعشرية فى الذاكرة مباشرة . شكل (3-7) يبين خريطة تدفق أو مخطط سير flow chart لخطوات حل أى مشكلة باستخدام المعالج .

السؤال الآن فى أى مكان فى الذاكرة سنضع البرنامج ؟

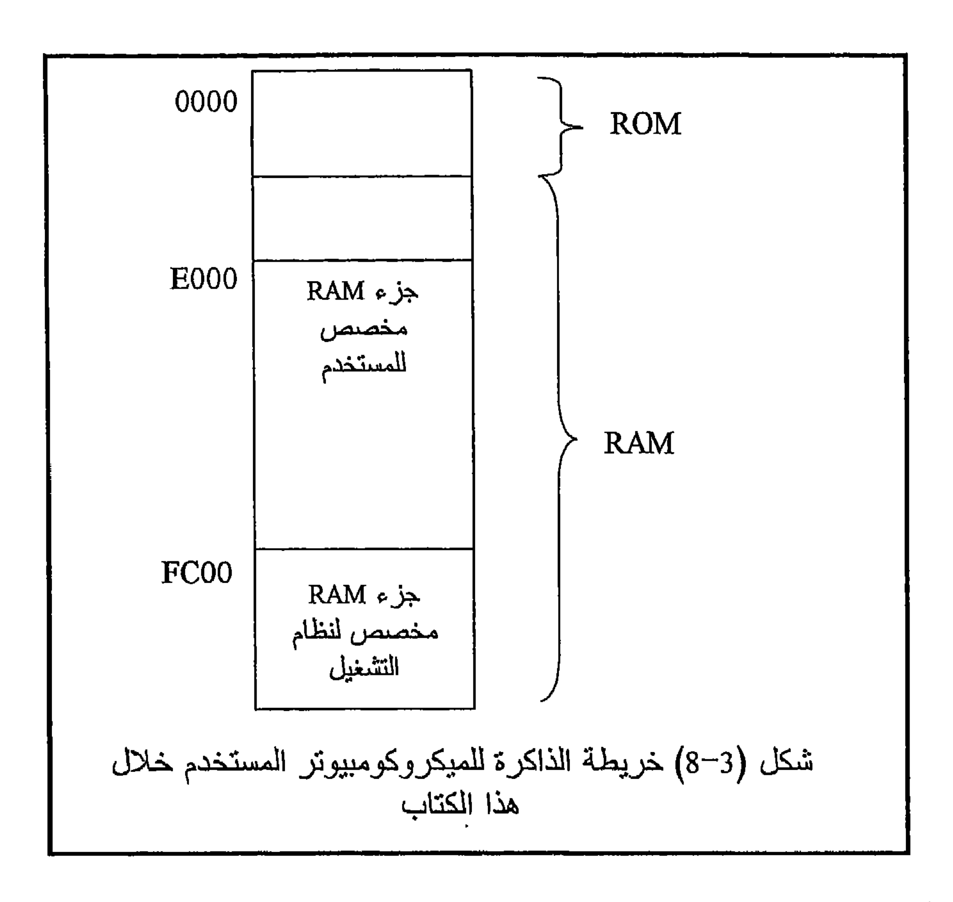
للإجابة عن هذا السؤال يجب أن نتصور الذاكرة الخاصة بالجهاز أو الميكروكومبيوتر الذى نستخدمه وقد قسمت إلى ثلاثة أجزاء كالموضحة فى شكل (3-8) . الجزء الأول منها هو ROM أو ذاكرة القراءة فقط وهذه كما ذكرنا فى الفصل الأول لا يمكن للمستخدم أن يسجل فيها أى شىء . الجزء الثانى من الذاكرة هى RAM وهى الجزء الذى يمكن للمستخدم أن يتعامل معه ويجب أن نعلم أن نظام التشغيل الخاص بالجهاز الذى نستخدمه يحجز جزءا من هذه RAM للاستعمال الخاص به والجزء المتبقى يمكن للمبرمج أن يستخدمه . لذلك يجب قبل أن تبدأ فى كتابة برنامجك بلغة الأسمبلى وإدخاله فى الذاكرة أن تعرف أين يقع جزء RAM الخاص بنظام التشغيل حتى لا يتداخل البرنامج الخاص بك معه. إن ذلك يتطلب إلقاء نظرة على ما يسمى بخريطة الذاكرة الخاصة بالميكروكومبيوتر الذى تستخدمه . هذه الخريطة تعتبر شكلا توضيحيا يبين الذاكرة بأكملها من الأول حتى آخر بايت وقد قسمت إلى أجزاء مع التعريف بكل جزء فيما يستخدم وهل هو مشغول أم لا . الآن وقد عرفت الجزء من RAM الذى يمكنك أن تضع فيه برنامجك يجب عليك كمبرمج أن تقسم هذا الجزء إلى

جزأين أيضا ، أحدهما تكتب فيه البرنامج والآخر تخصصه للبيانات التى يحتاجها أو يخرجها البرنامج .



شكل (3. 7) خريطة تدفق لخطوات برمجة أى مسألة لأى معالج

إن عملية تقسيم الذاكرة إلى جزء للبرنامج وآخر للبيانات عملية تعتمد على المبرمج بالدرجة الأولى وعلى البرنامج أيضا ، فقد يكون البرنامج لا يحتاج إلى بيانات أو لا يخرج بيانات على الإطلاق ، فى هذه الحالة فإن كل RAM ستكون للبرنامج ، وقد يكون البرنامج ينتج أو يحتاج للكثير من البيانات ، فى هذه الحالة يجب حجز جزء كاف لهذه البيانات .



شكل (3-8) خريطة الذاكرة للميكروكومبيوتر المستخدم خلال هذا الكتاب

مثال 3. 1

لدينا مجموعة من الأرقام وليكن عددها 50H رقما ، والمطلوب استخراج أكبر عدد فى هذه المجموعة . أين سنكتب البرنامج ؟ وأين سنكتب البيانات (الخمسين رقما) ؟

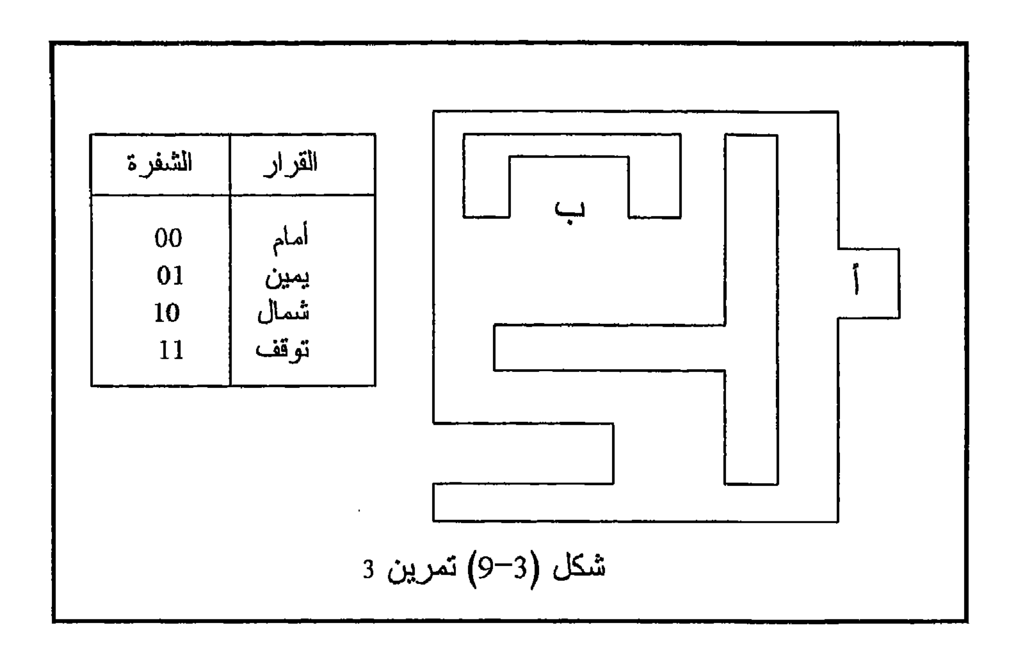
كما نرى من شكل (3-8) فإن جزء RAM المخصص للمستخدم فى الميكروكومبيوتر الذى نستخدمه يبدأ من العنوان E000H وينتهى عند العنوان FC00H . هذه المساحة من RAM يجب أن نقسمها إلى جزء للبرنامج وجزء للبيانات . البيانات المطلوبة هى 5 0H رقما ويمكن لنا أن نضعها ابتداء من العنوان E100H حتى العنوان E150H . أما البرنامج فيمكن لنا أن نكتبه مثلا ابتداء من العنوان E000H على أساس أن البايتات التى سيحتاجها البرنامج نفسه لن يصل عددها إلى 100H بايت بأى حال من الأحوال وإلا لو زاد عددها عن ذلك فسوف تتداخل مع البيانات عند العنوان E100H وفى هذه الحالة يجب أن نزيح البيانات بعيدا أو نكتب البرنامج بعد البيانات .

عملية تقسيم الذاكرة الموضحة فى شكل (3–8) يسمى فى مصطلحات الحاسبات بخريطة الذاكرة حيث يبين هذا الشكل جميع الذاكرة الملحقة بالجهاز الذى تستخدمه وفيما يستخدم كل جزء منها ، وهذه الخريطة يجب على أى مستخدم للغة الأسمبلى أن يتعرف عليها جيدا بالنسبة للحاسب الذى سيستخدمه ، والتقسيم الموضح فى شكل (3–8) سيكون هو التقسيم الذى سنتبعه خلال هذا الكتاب وبالطبع فإن هذه الخريطة تختلف باختلاف الجهاز المستخدم فيجب مراعاة ذلك .

3–9 تمارين

1. ما هو الأمر ؟
2. ما هو البرنامج ؟
3. تخيل أنك تركب سيارة آلية تريد الانتقال بها من المكان (أ) إلى المكان (ب) كما فى شكل (3. 9) هذه السيارة عند كل تقاطع لابد أن تأخذ قرارا من أربعة تحدد اتجاه حركتها كما فى الجدول الموضح فى نفس الشكل . اكتب برنامجا لهذه السيارة بالقرارات ثم بالشفرات الثنائية بحيث عندما تنفذه تنتقل السيارة من المكان (أ) إلى المكان (ب) فى الشكل .
4. هل يمكنك إعطاء اسم للغة الشفرات الثنائية المكتوب بها البرنامج السابق ؟
5. ماذا يحدث لو جعلنا شفرة القرار تتكون من 3 بتات بدلا من 2 ؟ بالطبع فى هذه الحالة سيكون هناك إمكانية لعدد أكبر من القرارات يصل إلى 8 ، افترض هذه القرارات من عندك محاولا تطوير هذه السيارة ؟
6. اشرح باستخدام خريطة تدفق كيف يقوم المعالج بتنفيذ أى برنامج ؟
7. ما هى عيوب كتابة البرامج بالشفرات الثنائية ؟
8. أعد كتابة البرنامج الموجود فى شكل (3–1) مستخدما النظام الثمانى ؟ كم عدد ضربات المفاتيح التى ستنفذها لكى تكتب البرنامج مستخدما هذا النظام ؟ ما نوع المحمل loader الذى ستحتاج إليه فى هذه الحالة ؟ أيهما أفضل فى الكتابة وسهولة استخراج الأخطاء ، النظام الثمانى أم النظام الستعشرى ؟
9. اشرح فائدة استخدام الأسمبلر ؟
10. ما هى عيوب البرمجة باستخدام لغة التجميع مقارنة باللغات ذات المستوى العالى ؟
11. اشرح الفرق بين الأسمبلر والمترجم ؟
12. هل لغة C من اللغات التى تعتمد على الماكينة ؟
13. مطلوب كتابة برنامج يجمع 20 رقما مخزنة فى الذاكرة ، ارسم خريطة للذاكرة تبين عليها أين ستكتب البرنامج وأين ستوضع البيانات ؟

14. حاول الحصول على خريطة الذاكرة للميكروكومبيوتر الــذى تتعــامل معــه وادرسها وطبق عليها المسألة السابقة ؟



القرار	الشفرة
أمام	00
يمين	01
شمال	10
توقف	11

شكل (3–9) تمرين 3

الفصل الرابع

برمجة المعالج Intel 8085

Programming The Intel 8085 Microprocessor

4-1 مقدمة

لبرمجة أى معالج لابد من دراسة مجموعة الأوامر الخاصة به ولكـــى نسـهل دراسة هذه الأوامر سنقوم بتقسيمها إلى مجموعات من حيث الوظيفة التى يؤديـها كل أمر وسندرس بالتفصيل فى كل مجموعة بعض الأوامر الكثيرة الاستخدام مـع التمثيل ببعض الأمثلة ، على أننا سنعرض فى نهاية الفصل لجداول تحتوى علـى جميع أوامر الشريحة موضوعة فى مجموعات ثم سنعرض أيضا جدولا يحتـوى هذه الأوامر مرتبة أبجديا مع نبذة بسيطة عن وظيفة كل أمر وشفرته .

4-2 مجموعة أوامر الانتقال Transfer instructions

يقوم أى أمر من أوامر هذه المجموعة بنقل معلومة من مكان لآخر حيث المكـان الذى تخرج منه المعلومة يسمى بالمصدر Source وسنرمز له بالرمز sss وهـذا المكان قد يكون مسجلا داخل شريحة المعالج وقد يكون مكانا من أماكن الذاكرة . وأما المكان الذى ستذهب إليه المعلومـة فسـوف نسـميه الـهدف Destination وسنرمز له بالرمز ddd وهذا المكان أيضا قد يكون مسجلا داخل شريحة المعـالج وقد يكون بايت من بايتات الذاكرة كما سنرى . شكل (4-1) يبين أهـم الأوامـر الموجودة فى مجموعة أوامر الانتقال الخاصة بالشريحة 8085 والتى تهمنا فـى هذه المرحلة من دراسة لغة الأسمبلى .

MOV
MVI
LXI
LDA
STA
LHLD
SHLD

شكل (4-1) بعض أوامر الانتقال الكثيرة الاستخدام للشريحة 8085

4-2-1 الأمر MOV

الصورة العامة لهذا الأمر هى :

MOV ddd,sss

مسجل ddd ← مسجل sss

ومعنى هذا الأمر انقل أو حرك (وهذا هو ترجمة كلمة move) المعلومة الموجودة فى المصدر sss إلى الهدف ddd ، لاحظ أن المعالج عندما يقوم بنقل معلومة فإنه ينقل صورة منها فقط أما أصل المعلومة فيظل فى المصدر ولا يتغير . الصورة العامة لشفرة هذا الأمر الثنائية يمكن كتابتها كما يلى :

01dddsss

حيث sss تستبدل بشفرة مصدر المعلومة سواء كانت مسجلا أم ذاكرة و ddd تستبدل أيضا بشفرة الهدف الذى ستلجأ إليه المعلومة سواء كان مسجلا أم ذاكرة. لاحظ أن هذا الأمر يتكون دائما من بايت واحدة . راجع شفرات المسجلات فى الفصل الثانى ، جدول 2-1 ، وانظر المثال التالى :

مثال 4-1

1. الأمر MOV A,B

شفرته الثنائية هى 01111000

شفرته الستعشرية هى 78H

هذا الأمر سينقل محتويات المسجل B (صورة منها فقط) إلى المسجل A لاحظ أن المسجل B هو المصدر sss ولذلك استبدلناه بشفرته الثنائية وهى 000 والمسجل A هو الهدف ddd واستبدلناه بشفرته الثنائية 111 لتصبح شفرة الأمر الثنائية هى 01111000 أو 78H فى النظام الستعشرى ، ولمزيد من الأمثلة إليك ما يلى :

2. الأمر MOV L,D

شفرته الثنائية هى 01101010

شفرته الستعشرية هى 6AH

3. الأمر MOV C,C

شفرته الثنائية هى 01001001

شفرته الستعشرية هى 49H

هذا الأمر ينقل محتويات المسجل C إلى نفسه وهذا يكافئ تماما ، لا تعمل شيئا . مثل هذه الأوامر التى لا تعمل شيئا لها أهمية كبيرة فى الكثير من التطبيقات كما سيأتى فيما بعد .

فى جميع الأوامر السابقة كنا ننقل المعلومة من مسجل إلى مسجل آخر ، ماذا لو أردنا نقل معلومة من مسجل إلى الذاكرة أو العكس . المثال التالى سيوضح ذلك .

مثال 4-2

1. الأمر MOV M,A

شفرته الثنائية هى 01110111

شفرته الستعشرية هى 77H

هذا الأمر ينقل محتويات المسجل A وهو مصدر المعلومة إلى الذاكرة M وهى الهدف الذى ستذهب إليه المعلومة ، لاحظ أن M استبدلت بالشفرة 110 كما فى جدول 2-1 . السؤال الآن هو : في أى مكان أو فى أى عنوان فى الذاكرة ستذهب محتويات المسجل A ؟ فى جميع الأوامر التى تتعامل مع الذاكرة بهذا الشكل يكون العنوان موجودا فى زوج المسجلات HL ، أى أن محتويات المسجل A ستذهب إلى بايت الذاكرة التى يوجد عنوانها فى المسجلين H و L . لاحظ أن هذه الطريقة هى ما سنسميها بطريقة التعامل غير المباشر مع الذاكرة عندما سنصنف طرق التعامل مع الذاكرة فى نهاية هذا الفصل .

2. الأمر MOV B,M

01000110

46H

هذا الأمر سينقل محتويات بايت الذاكرة التى يوجد عنوانها فى المسجلين H و L إلى المسجل B .

4-2-2 الأمر MVI

هذا الأمر معناه "انقل المعلومة الفورية" أى Move the Immediate data والصورة العامة له هى :

MVI ddd,data8

ddd ⟵ data8

هذا لأمر يضع المعلومة المكونة من ثمانية بتات (data8) فى الهدف ddd . الهدف ddd قد يكون مسجلا أو الذاكرة M كما سنرى في الأمثلة . هذا الأمر يتكون دائما من اثنتين من البايتات . واحدة هى شفرة الأمر operation code واختصارا op code والبايت الأخرى هى المعلومة (data8) . وعلى ذلك ستكون الصورة العامة للشفرة الثنائية لهذا الأمر كالتالى :

00ddd110

data8

حيث ddd تستبدل بشفرة المسجل أو الذاكرة M المراد وضع المعلومة فيها .

مثال 4-3

1. MVI B,53H

الشفرة الثنائية هى : 00000110

01010011

الشفرة الستعشرية هى : 06H
53H

هذا الأمر سيضع المعلومة الفورية أو الثابت 53H فى المسجل B . نفضـــــل أن نسمى مثل هذه المعلومة بالمعلومة الفورية لأنها ليس لها مصدرا وإنما مصدرهـا هو المستخدم نفسه ، ومن هنا كان الحرف I فى الأمر وهو اختصـــــارا لكلمـــة Immediate أو فورى وسوف يصادفنا أوامر أخرى تحتوى الحـــرف I وكلـــها تتعامل مع معلومات فورية أو ثوابت بهذا الشكل .

من الممكن تحميل معلومة فورية Immediate فى مكان ما فى الذاكـــرة بحيـــث يكون عنوان هذا المكان فى المسجلين HL كالتالى :

.2 MVI M,data8

والشفرة الثنائية لهذا الأمر ستكون كالتالى :

00110110
data8

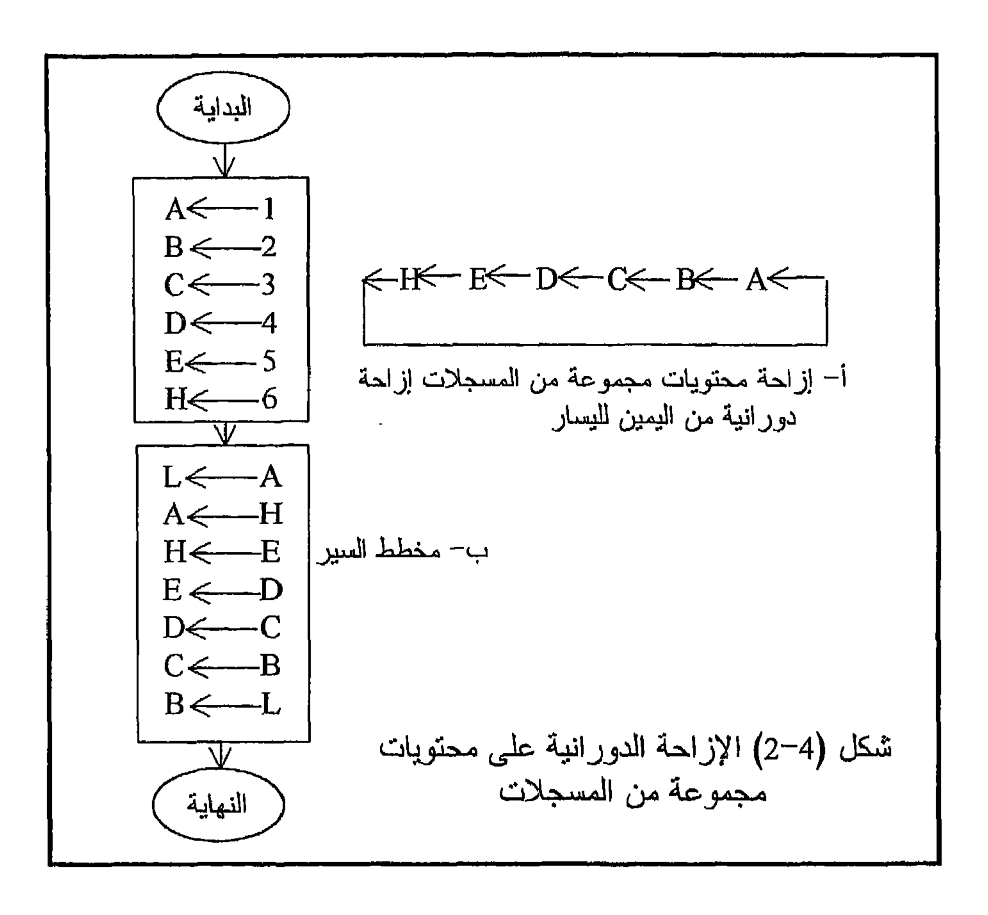
المثال التالى يعتبر تدريبا جيدا على الأمرين MOV و MVI

مثال 4-4

المطلوب تحميل المسجلات H, E, D, C, B, A بالمعلومات الفورية التالية : 06, 05, 04, 03, 02, 01 على التوالى ، ثم بعد ذلك يتم عمل إزاحة دورانية لـــهذه المحتويات كما هو مبين فى شكل (4-2أ) بحيث أن محتويات المسجل A تذهب إلى المسجل B ومحتويات B تذهب إلى C وهكـــذا إلـــى أن تذهـــب محتويـــات المسجل H إلى المسجل A دون فقد محتويات أى مسجل . شكل (4-2ب) يوضح مخطط السير لهذا البرنامج ، وشكل (4-3) يبين الشفرات الأســـمبلى والثنائيـــة والستعشرية للبرنامج . لتنفيذ هذا البرنامج يمكنك أن تفتح ال RAM عند العنوان E000 وتبدأ فى إدخال البرنامج إما بالشفرات الثنائية أو الشفرات الستعشـــرية ، وأما إذا كان الميكروكومبيوتر الذى تستخدمه به الأسمبلر الخـــاص بالشـــريحة 8085 فيمكنك الدخول فيه وكتابة البرنامج باستخدام شفرات الأسمبلى وســـنترك تفاصيل عملية إدخال البرنامج على الجهاز لأنها تختلف على حسب الإمكانيـــات ومن شخص لآخر . بعد الانتهاء من إدخال البرنامج يمكنـــك تنفيـــذه باســـتخدام الأمر GO E000 أو من الأفضل تنفيذه بطريقة الخطوة خطوة حيث يمكنك فـــى هذه الحالة متابعة البيانات أثناء انتقالها من مسجل لآخر .

عند استخدام الأمر MOV M,C مثلا لنقل محتويات المسجل C إلى الذاكـــرة أو الأمر MVI M,33 لتحميل المعلومة الفورية 33 فى الذاكرة ذكرنـــا أن عنـــوان الذاكرة الذى سيتم التعامل معه يكون دائما فى المسجلين HL . السؤال الآن كيف

نضع هذا العنوان فى المسجلين H و L؟ الإجابة عن هذا السؤال توجد فى الأمر LXI .



شكل (4-2) الإزاحة الدورانية على محتويات مجموعة من المسجلات

4-2-3 الأمر LXI

الصورة العامة لهذا الأمر هى:

LXI rp,data16

rp زوج مسجلات ← data16

الشفرة الثنائية للأمر كالتالى :

00rp0001

البايت ذات القيمة الصغرى من المعلومة data16

البايت ذات القيمة العظمى من المعلومة data16

حيث يقوم هذا الأمر بتحميل زوج المسجلات الذى توضع شفرته بدلا من rp (فى البايت الأولى للأمر) بالمعلومة الفورية data16 أى المكونة من 16 بت والتى توضع فى الإثنين بايت التاليتين . حرف I الموجود فى صورة الأمر له

نفس الدلالة التى عرفناها مسبقا والتى تعنى فورى أى Immediate . لاحظ أن المعلومة مكونة من 16 بت وشفرة الأمر op code ستكون بايت واحدة ، لذلك فإن هذا الأمر يتكون دائما من ثلاث بايتات كما سنرى فى الأمثلة . لاحظ أيضا أن هناك أربعة أزواج من المسجلات فقط يمكن استخدامها مع هذا الأمر وهى الإزواج SP , HL, DE, BC والتى سبق أن ذكرنا شفراتها فى جدول 2-1 . لذلك فإن البايت ذات القيمة العظمى من المعلومة ال16 بت تذهب دائما إلى المسجل ذى القيمة العظمى من الزوج وهو المسجل B أو D أو H أو النصف الأعلى من المسجل SP والبايت ذات القيمة الصغرى تذهب إلى المسجل ذى القيمة الصغرى من الزوج وهو المسجل C أو E أو L أو النصف الأول من المسجل SP .

العناوين	شفرات أسمبلى	شفرات ثنائية	شفرات ستعشرية
E000 E001	MVI A,01	00111110 00000001	3E 01
E002 E003	MVI B,02	00000110 00000010	06 02
E004 E005	MVI C,03	00001110 00000011	0E 03
E006 E007	MVI D,04	00010110 00000100	16 04
E008 E009	MVI E,05	00011110 00000101	1E 05
E00A E00B	MVI H,06	00100110 00000110	26 06
E00C	MOV L,A	01101111	6F
E00D	MOV A,H	01111100	7C
E00E	MOV H,E	01100011	63
E00F	MOV E,D	01011010	5A
E010	MOV D,C	01010001	51
E011	MOV C,B	01001000	48
E012	MOV B,L	01000101	45

شكل (4-3) برنامج الإزاحة الدورية

مثال 4-5

1. شفرة الأسمبلى LXI H,2C3A

الشفرة الثنائية 00100001

00111010

00101100

الشفرة الستعشرية 21

3A

2C

يقوم هذا الأمر بتحميل زوج المسجلات HL بالمعلومة الفورية 2C3A بحيث ستذهب البايت ذات القيمة العظمى وهى 2C إلى المسجل H والبايت ذات القيمة الصغرى وهى 3A إلى المسجل L . بنفس الطريقة يمكن تحميل الأزواج الأخرى من المسجلات . كما رأينا فإن هذا الأمر مهم جدا عند التعامل مع الذاكرة حيث عنوان المكان المراد التعامل معه يوضع فى المسجلين H و L باستخدام هذا الأمر .

مثال 4-6

حمل مكان الذاكرة E100 بالمعلومة 66H . أحد الطرق لكى يتم ذلك هى أن نقوم بتحميل العنوان E100 فى زوج المسجلات HL ثم نستخدم الأمر MVI كما يلى :

E000, E001, E002 LXI H,E100

E003, E004 MVI M,66H

لاحظ أننا استخدمنا هنا الشفرات الأسمبلى فقط ولاحظ أيضا أن الأمر LXI يشغل ثلاث بايتات كما ذكرنا وهى البايتات E000, E001, E002 .

إن الطريقة السابقة فى التعامل مع الذاكرة والتى تستخدم زوج المسجلات HL كوسيط يحمل العنوان المراد التعامل معه تعتبر بل وتسمى بالطريقة غير المباشرة فى التعامل مع الذاكرة حيث أن المعالج قبل أن يذهب إلى الذاكرة سواء للقراءة أو للكتابة لابد وأن يمر أولا على زوج المسجلات HL ليعرف منهما العنوان الذى سيذهب إليه . إن ذلك تماما مثلما أنك تقول لصديقك محمد ياأخى يامحمد وأنت مسافر إلى الرياض خذ هذه الرسالة وأعطها لأخى أحمد ولكنى لا أعرف عنوانه فرجاء أن تذهب إلى أخى الأكبر محمود وهو يسكن معنا هنا فتعرف منه عنوان أحمد قبل أن تسافر إلى الرياض . صديقك محمد فى هذا المثال يمثل المعالج الذى سيقوم بالتنفيذ وأما أخوك الأكبر محمود فيمثل المسجلين HL الذين عندهما عنوان أخيك أحمد الذى يمثل بايت الذاكرة المراد التعامل معها . نعيد التأكيد هنا أن زوج المسجلات HL فقط هما اللذان يحتويان عنوان المكان المراد التعامل معه فى مثل هذه الأوامر .

هناك طريقة أخرى للتعامل مع الذاكرة وهى الطريقة المباشـــرة حيــث يحتـــوى الأمر نفسه على عنوان مكان الذاكرة المراد التعامل معه ، وإن ذلك مثلما تقــول لصديقك محمد يأخى يا محمد خذ هذه الرسالة وأنت مسافر إلى الرياض وأعطها إلى أخى أحمد ولا تنسى أن عنوان أخى أحمد مكتوب على الرسالة هذه المــرة . الأمران الذان يقومان بهذه المهمة من قائمة أوامر الشريحة 8085 هما الأمـــران STA و LDA كما يلى :

4-2-4 الأمران STA و LDA

الأمر STA يعنى خزن محتويات المركم STore Accumulator والأمـــر الثـــانى يعنى حمل المركم LoaD Accumulator والصورة العامة للأمرين هى :

STA addr

محتويات المسجل A ⟵ العنوان (addr)

LDA addr

محتويات العنوان (addr) ⟵ المسجل A

والصورة الستعشرية للأمر STA هى :

32

البايت ذات القيمة الصغرى من العنوان Addr

البايت ذات القيمة العظمى من العنوان Addr

والصورة الستعشرية للأمر LDA هى :

3A

البايت ذات القيمة الصغرى من العنوان Addr

البايت ذات القيمة العظمى من العنوان Addr

الأمر الأول STA سيقوم يتخزين محتويات مسجل التراكم A فى مكان الذاكـــرة الذى عنوانه فى (الإثنين بايت) التاليتين لشفرة الأمر نفسه وأما الأمر LDA فإنـــه يقوم بتحميل مسجل التراكم A بمحتويات مكان الذاكرة الذى عنوانه فى (الإثنيـن بايت) التاليتين لشفرة الأمر نفسه . هناك ملاحظتان هامتان على هذين الأمرين : الأولى هى أن هذين الأمرين لا يتعاملان إلا مع مسجل التراكم A فقط ، فـــأنت مثلا إن أردت أن تخزن محتويات المسجل B فى أى مكان فى الذاكـــرة بـــهذه الطريقة المباشرة فلابد وأن تنقل محتويات المسجل B إلى المسجل A أولا ثـــم تستخدم الأمر STA ، وكذلك الحال إن أردت أن تحمل أى مسجل وليكن C مثلا بمحتويات أى مكان فى الذاكرة بالطريقة المباشرة فعليك باستخدام الأمـــر LDA الذى يضع محتويات الذاكرة فى مسجل التراكم ثم تقوم أنت بنقل هذه المحتويـات من مسجل التراكم إلى أى مسجل آخر وليكن C كما ذكرنا . الملاحظة الثانيـــة هى أنه طالما أن هذين الأمرين يحتويان على عنوان فلابد وأن يتكون كل منـــهما من ثلاث بايتات ، واحدة هى شفرة الأمر ، أى 32 فى حالة الأمـــر STA و 3A

فى حالة الأمر LDA وأما (الإثنين بايت) الأخرى فتحتوى العنوان المراد التعامل معه كما فى الصورة العامة للأمرين . تذكر أن أى عنوان يتكون دائما من 16 بت أى 2 بايت .

مثال 4-7

E000 E001 E002 STA E100
E003 E004 E005 LDA E101

الأمر الأول سيخزن محتويات المسجل A فى المكان E100 فى الذاكرة والأمر الثانى سيحمل المسجل A بمحتويات المكان E101 من الذاكرة ، لاحظ أنه فى ذاكرة البرنامج كل أمر من الأمرين يشغل ثلاث بايتات .

القرار باستخدام أى واحدة من الطريقتين (المباشرة أو غير المباشرة) فى عملية البرمجة يتوقف على التطبيق الذى يستخدم فيه هذا البرنامج وسنرجىء هذا الموضوع قليلا إلى أن ندرس بعض الأوامر الأخرى وعندها سنوضح أن هناك برامج يفضل فيها استخدام الطريقة المباشرة وأخرى لابد فيها من استخدام الطريقة غير المباشرة .

4-2-5 الأمران LHLD و SHLD

معناهما load H,L Direct أى حمل المسجلين H, L مباشرة ، و Store H,L Direct أى خزن المسجلين H, L مباشرة وكلا الأمرين كما نرى تأثيره عكس الآخر . الصورة العامة لهذين الأمرين هى :

LHLD addr

محتويات (addr+1) ⟵ H ، محتويات (addr) ⟵ L

SHLD addr

محتويات H ⟵ (addr+1) ، محتويات L ⟵ (addr)

الصورة الستعشرية للأمر LHLD هى :

2A

البايت ذات القيمة الصغرى للعنوان addr

البايت ذات القيمة العظمى للعنوان addr

الصورة الستعشرية للأمر SHLD هى :

22

البايت ذات القيمة الصغرى للعنوان addr

البايت ذات القيمة العظمى للعنوان addr

حيث يقوم الأمر الأول LHLD بتحميل المسجلين H و L بمحتويات العنوان addr والذى يليه فى الذاكرة ، أى أن محتويات العنوان addr تذهب إلى المسجل L ومحتويات العنوان الذى يليه addr+1 تذهب إلى المسجل H . الأمر SHLD

يقوم بالعملية العكسية للأمر LHLD حيث يقوم بتخزين محتويات المسجلين H و L فى العنوان addr والذى يليه بحيث تذهب محتويات المسجل L إلى العنوان addr ومحتويات المسجل H إلى العنوان addr+1 .

مثال 4-8

1. افترض الوضع التالى فى المسجلين HL ومكانى الذاكرة E100 و E101 :

H	L
89	76

E100	FF
E101	FF

بعد تنفيذ الأمر SHLD E100 سيصبح الوضع كما يلى :

H	L
89	76

E100	76
E101	89

2. افترض الوضع التالى فى المسجلين HL ومكانى الذاكرة E100 و E101 :

H	L
89	76

E100	3A
E101	B5

بعد تنفيذ الأمر LHLD E100 سيصبح الوضع كالتالى :

H	L
B5	3A

E100	3A
E101	B5

إلى هنا سنكتفى بهذه المجموعة من أوامر الانتقال ولمزيد من المعرفة بباقى أوامر الإنتقال انظر الأشكال الموجودة فى آخر هذا الفصل والتى تحتوى على نبذة مختصرة عن كل أمر .

4-3 تمارين

1. اكتب برنامجا يضع الأرقام الستعشرية 1 إلى A فى الذاكرة E201 إلى E20A.

2. اكتب برنامجا يضع الرقم 77 فى المسجل D والرقم B4 فى المسجل C ثم يقوم باستبدال محتويات كل من المسجلين ، أى أن محتويات المسجل D تذهب إلى المسجل C ومحتويات المسجل C تذهب إلى المسجل D دون أن تفقد أيا من محتويات المسجلين ، هذه العملية تسمى Swapping بمعنى استبدال أو مقايضة .
3. نفذ عملية الاستبدال السابقة فى المسألة رقم 2 ولكن هذه المرة على محتويات البايت E100 والبايت E101 .
4. اكتب برنامجا للإزاحة الدورانية كالموجود فى مثال 4-4 ولكن هذه المرة على محتويات الذاكرة E100 و E101 و E102 و E103 و E104 و E105 .

ملاحظات : يجب عمل مخطط سير لكل برنامج ثم تترجم هذه الخريطة إلى برنامج بلغة الأسمبلى مع الشفرات الثنائية والستعشرية وتحديد العناوين لكل أمر وذاكرة البرنامج وذاكرة البيانات لكل برنامج ، أى أن كل برنامج يحل بنفس طريقة حل المثال رقم 4-4 وحتى تكتمل الفائدة بالذات عند هذا المستوى يجب أن تحمل هذه البرامج بالشفرات الثنائية أولا ثم الشفرات الستعشرية ثم شفرات الأسمبلى .

4-4 مجموعة أوامر الحساب
Arithmatic Instructions

ADD
SUB
ADI
SUI
ADC
SBB
INR
INX
DCR
DCX

شكل (4-4) مجموعة أوامر الحساب

سندرس فى هذا الجزء بعض الأوامر التى تقوم بإجراء العمليات الحسابية الأولية وهى الجمع والطرح . كما علمنا من قبل فإن مسجل التراكم لابد وأن يكون طرفا فى أى عملية من هذه العمليات كما أن نتيجة هذه العملية سواء كانت جمعا أو طرحا تكون دائما موجودة فى مسجل التراكم A . هناك أيضا خاصية مهمة

فى هذه المجموعة من الأوامر غير موجودة فى الأوامر الأخرى وهى أن أوامر الحساب (ومثلها أيضا أوامر المنطق) عندما يتم تنفيذ أى أمر منها فإن جميع الأعلام الموجودة في مسجل الحالة SR تتأثر بنتيجة هذه العملية الحسابية أو المنطقية . راجع مسجل الحالة ومحتوياته ومتى يكون أى علم من الأعلام يساوى صفرا أو واحدا وذلك فى الفصل الثانى حيث أن جميع الأعلام الموجودة فى مسجل الحالة فى المعالج 8085 هى نفسها تماما التى شرحت فى هذا الجزء (2-4-4) . شكل (4-4) يبين مجموعة الأوامر التى سندرسها هنا حيث فى جميع الأمثلة التى سنستعين بها فى هذا الجزء سنكتفى باستخدام شفرات الأسمبلى فقط على أساس أنه تم التدريب الكافى على الشفرات الثنائية والستعشرية مع مجموعة أوامر الانتقال ومن يريد الاستزادة فى التدريب يمكنه الإستمرار فى ذلك مستعينا بالشفرات الستعشرية التى سنذكرها مع الصورة العامة لكل أمر وكذلك فى الأشكال الموجودة فى نهاية هذا الفصل .

4-4-1 الأمران ADD و SUB

حيث ADD بمعنى إجمع و SUB هى أختصار كلمة SUBtract بمعنى اطرح ، والصورة العامة لأمر الجمع ADD هى :

ADD reg
A⟸A + reg

حيث سيقوم هذا الأمر بجمع محتويات المسجل reg أو مكان الذاكرة M الذى عنوانه فى المسجلين HL على محتويات مسجل التراكم A وستوضع نتيجة الجمع فى المسجل A مع التأثير على جميع الأعلام . الشفرة الثنائية العامة لهذا الأمر هى 10000xxx حيث تستبدل xxx بشفرة المسجل الذى سيتم جمعه مع مسجل التراكم . كمثال على ذلك الأمر ADD B ستكون شفرته الثنائية هى 10000000 وشفرته الستعشرية هى 80H حيث استبدلت xxx بشفرة المسجل B وهى 000 .

الصورة العامة لأمر الطرح SUB هى :

SUB reg
A⟸ A - reg

يقوم هذا الأمر بطرح محتويات المسجل reg أو مكان الذاكرة M الذى عنوانه فى المسجلين HL من محتويات مسجل التراكم وسوف توضع نتيجة الطرح فى مسجل التراكم مع التأثير على جميع الأعلام . الشفرة الثنائية العامة لهذا الأمر هى 10010xxx حيث تستبدل xxx بشفرة المسجل المطلوب طرح محتوياته من مسجل التراكم . كمثال على ذلك الأمر SUB M ستكون شفرته الثنائية هى 10010110 وشفرته الستعشرية هى 96H .

مثال 4-9

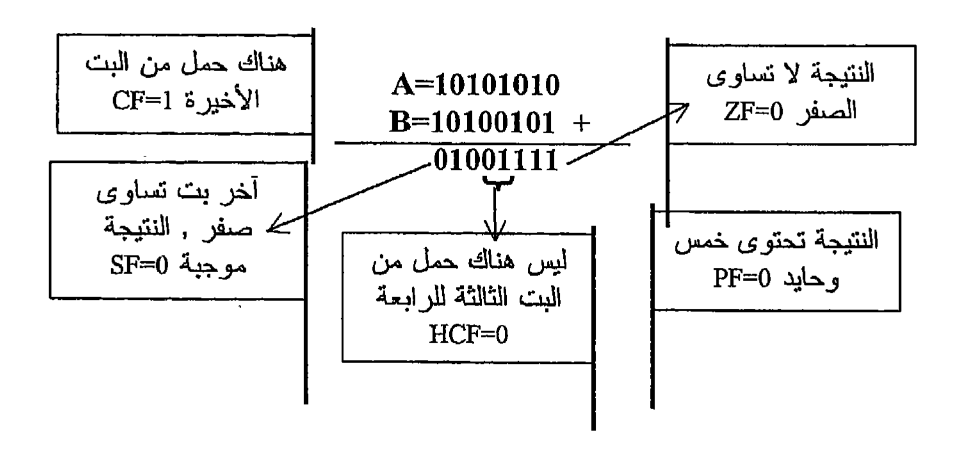
افترض المحتويات التالية للمسجلين A و B :



بعد تنفيذ الأمر ADD B ستصبح المحتويات كالتالى :



ولكى نرى كيف تتأثر الأعلام بنتيجة هذه العملية سنجرى عملية الجمـــع علـــى الشفرات الثنائية لكل من الرقمين كالتالى :



الآن افترض أننا نفذنا أمر الطرح SUB B على المحتويات الأولى للمســـجلين أى A=AAH و B=A5H فإنه بعد تنفيذ هذا الأمر ســـتصبح محتويـــات المســـجلين كالتالى:

B	A
A5	05

ولكى نرى كيف تمت عملية الطرح وكيف تأثرت الأعلام سنجرى عملية الطـرح على الشفرات الثنائية لمحتويات للمسجلين A و B . كما نعلم فإن عملية الطـرح الثنائى يتم تحويلها إلى عملية جمع حيث سنجمع محتويات المسجل A مع المتمم الثنائى لمحتويات المسجل B (انظر الملحق الأول فى نهاية الكتاب لمراجعة عمليات الجمع والطرح الثنائى). المتمم الثنائى لمحتويات المسجل B(10100101) هو 01011011 وبذلك تصبح عملية الطرح عملية جمع كالتالى :

A = 10101010

+ 01011011 المتمم الثنائى لمحتويات المسجل B

00000101

بالنظر لهذه النتيجة ستكون الأعلام كالتالى :

- طالما أن البت الأولى على الأقل لا تساوى صفرا فالنتيجة لا تساوى صفرا ويكون علم الصفر يساوى صفرا أى ZF=0 .
- تحتوى النتيجة على عدد زوجى من الوحايد (اثنين) ، بذلك سيكون علم الباريتى يساوى واحدا أى PF=1 .
- هناك حمل من البت الثالثة إلى البت الرابعة لذلك فعلم الحمل النصفى يكون واحدا HCF=1 .
- آخر بت (رقم 7) تساوى صفرا لذلك فالنتيجة موجبة وعلم الإشارة يكون دائما مساويا لمحتويات آخر بت إذن SF=0 .
- المفروض أنه فى عملية الطرح يهمنا أن نعرف إذا كان هناك استلاف أم لا لأنه فى عملية الطرح لن يكون هناك حمل . بما أن عملية الطرح قد حولت إلى عملية جمع لذلك فإنه إذا كان هناك حمل فى عملية الجمع فإن ذلك يعنى أنه لن يكون هناك استلاف فى عملية الطرح ويكون العلم CF=0 وهى الحالة التى نحن بصددها الآن والعكس صحيح إذا لم يكن هناك حمل فى عملية الجمع .

2-4-4 الأمران ADI و SUI

الأمر الأول اختصارا ل Add Immediate أى اجمع المعلومة الفورية أو الثابت ، والأمر الثانى SUI اختصارا ل Subtract Immediate والتى تعنى أيضا اطرح المعلومة الفورية أو الثابت ، وقد أشرنا سابقا إلى أن الحرف I فى أى أمر يعنى التعامل مع معلومة فورية أو ثابت يعطى فى الأمر نفسه . الصورة العامة للأمر ADI هى :

ADI data8

A⟵A + data8

والشفرة الستعشرية العامة لهذا الأمر هى :

C6

data8

حيث يقوم هذا الأمر بجمع محتويات البايت الثانية فى الأمر مع محتويات مسجل التراكم A ويضع النتيجة فى المسجل A وعلى ضوء هذه النتيجة تتأثر جميع الإعلام . لاحظ أن هذا الأمر يتكون دائما من اثنين بايت ، واحدة هى شفرة الأمر والثانية هى البايت أو الرقم المطلوب جمعه مع المسجل A .

الصورة العامة للأمر SUI هى :

SUI data8
A⟵ A − data8

والشفرة الستعشرية لهذا الأمر هى :

D6
data8

حيث يقوم هذا الأمر بطرح محتويات البايت الثانية فى الأمر من محتويات مسجل التراكم A ويضع النتيجة فى المسجل A وعلى ضوء هذه النتيجة تتأثر جميع الأعلام . لاحظ أن هذا الأمر أيضا يتكون من اثنين بايت .

مثال 4-10

المطلوب هو جمع الثابت أو المعلومة الفورية 05 على محتويات المسجلات D, C, B . شكل (4-5) يبين مخطط السير وشفرات الأسمبلى لهذا البرنامج . ملاحظة مهمة فى هذا البرنامج هى أنه لكى نجمع الثابت مع أى مسجل فإننا نحضر أو ننقل محتويات المسجل أولا إلى مسجل التراكم A ثم نجمع الثابت مع المسجل A وبالطبع فإن النتيجة تكون فى المسجل A فنقوم بنقلها إلى المسجل الآخر ثانية. وهل يوجد حل آخر؟!

4-4-3 الأمران ADC و SBB

الأمر الأول يعنى Add with Carry أى اجمع مع الحمل ، والثانى يعنى Subtract with Borrow أى اطرح مع الاستلاف والصورة العامة للأمر ADC هى :

ADC reg
A⟵ A + CY + reg

حيث يقوم هذا الأمر بجمع محتويات المسجل reg مع محتويات علم الحمل CY (لاحظ أن علم الحمل يكون إما صفرا أو واحد) مع محتويات مسجل التراكم A والنتيجة توضع بالطبع فى المسجل A . الشفرة الثنائية لهذا الأمرهى :

10001xxx

حيث تستبدل ال xxx بشفرة المسجل المراد التعامل معه .

الصورة العامة للأمر SBB هى :

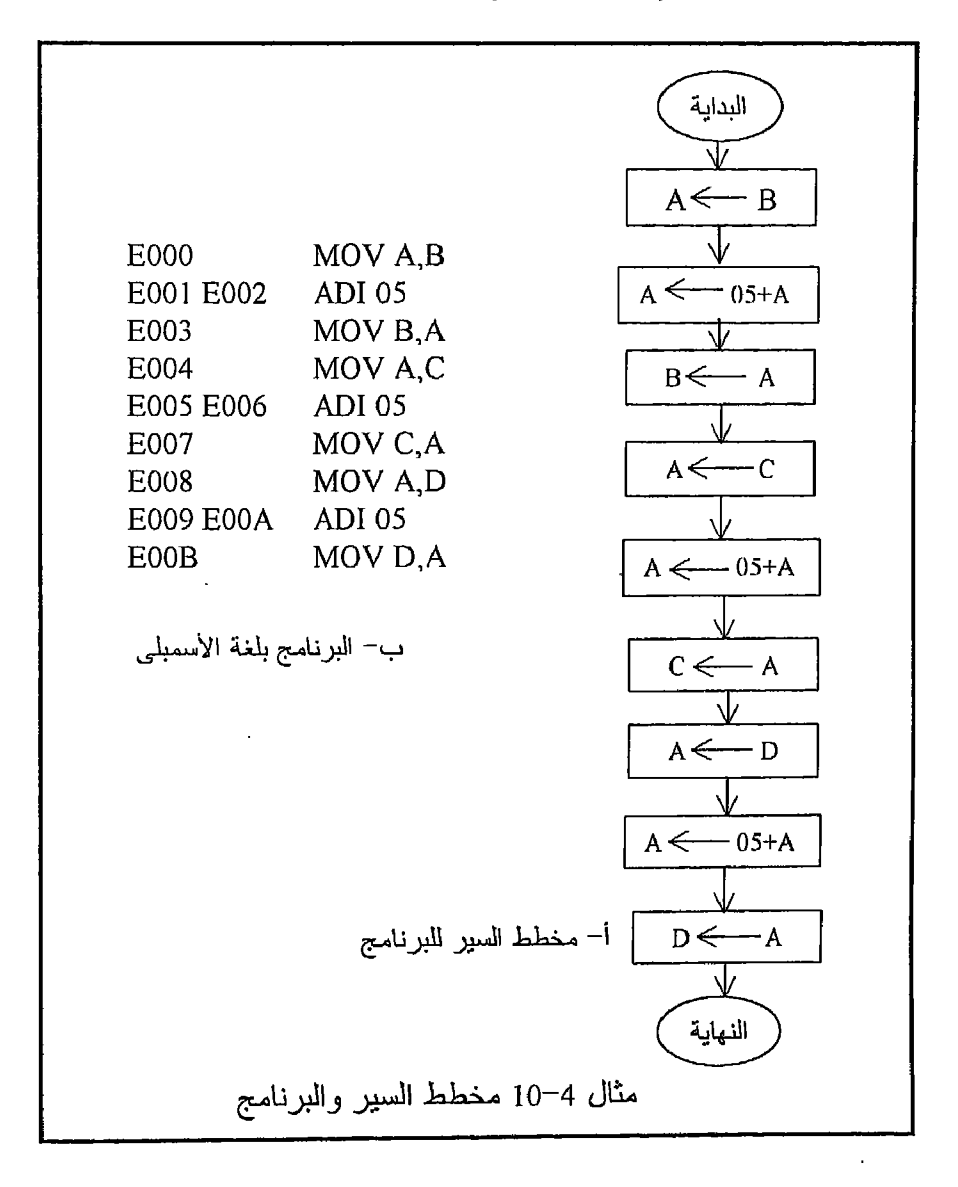
SBB reg

A⟵ A − CY − reg

حيث يطرح هذا الأمر من محتويات مسجل التراكم A محتويات المسجل reg ومحتويات علم الحمل CY مع ملاحظة أن المسجل A يكون هو المطروح منه دائما فى جميع أوامر الطرح . الشفرة الثنائية لهذا الأمر هى :

10011xxx

حيث تستبدل ال xxx بشفرة المسجل المراد التعامل معه .



مثال 4-10 مخطط السير والبرنامج

مثال 4-11

المطلوب جمع الرقمين 23F9H و 9A35H ووضع نتيجة الجمع فى أماكن الذاكرة E100 و E101 و E102 . المشكلة فى هذين الرقمين أن كلا منهما يشغل (اثنيـــن بايت) ونحن نعلم أن جميع أوامر الجمع تتعامل مع بايت واحدة فقط فما الحـــل ؟ الحل لهذه المشكلة ممكن بأن نضع الرقم الأول مثلا فى المسجلين B و C بحيـــث يحتوى المسجل B البايت ذات القيمة العظمى من الرقم وهى 23 ويحتوى المسجل C البايت ذات القيمة الصغرى وهى F9 وبنفس الطريقة نضع الرقم الثـــانى فـــى المسجلين D و E بحيث يحتوى المسجل D البايت 9A ويحتوى المسجل E البايت 35 . بعد ذلك سنجمع البايت ذات القيمة الصغرى فى كل من الرقمين أى المسجل C زائد المسجل E باستخدام الأمر ADD وسينتج عن ذلك نتيجة وحمل ، النتيجة نخزنها فى الذاكرة E100 ، وبعد ذلك نستخدم الأمر ADC لجمـــع البـــايت ذات القيمة العظمى فى الرقمين مع محتويات علم الحمل وهى الحمل الناتج من عمليـــة الجمع الماضية وسينتج عن ذلك نتيجة تخزن فى الذاكرة E101 وحمـــل ، هـــذا الحمل يخزن فى الذاكرة E102 كجزء من النتيجة . شـــكل (4-6) يبيـــن رســـما توضيحيا للحل وخريطة التدفق والبرنامج لهذا المثال .

قبل أن نترك الأمر ADC يجب أن نفهم جيدا متى يكون من الضرورى اســـتخدام الأمر ADC ؟ ومتى يكون من الضرورى عدم استخدامه ؟ فمثـــلا فـــى المثـــال السابق (4-11) كان من الضرورى عدم استخدام الأمر ADC فى عملية الجمـــع الأولى (E + C) ولكن يجب استخدم الأمر ADD خوفا من أن يكون علم الحمـــل CY به واحد من أى عملية سابقة ونحن لا ندرى فيجمع مع عملية الجمع الأولـــى وتكون النتيجة خاطئة . أما فى عملية الجمع الثانية (D + B + CY) فإنه لابد مـــن استخدام الأمر ADC لأننا نريد أن نأخذ قيمة علم الحمل CY فى الاعتبار .

4-4-4 الأمران INR و DCR

الأمر الأول يعنى Increment أى زد المحتويات بمقدار واحد ، و Decrement أى انقص المحتويات بمقدار واحد ، والصورة العامة للأمر INR هى :

INR reg

reg ⟵ 1 + reg

حيث يقوم هذا الأمر بجمع واحد على محتويات المسجل reg أو مكان الذاكـــرة M الذى عنوانه فى HL . الصورة الثنائية لهذا الأمر هى :

00xxx100

حيث xxx تستبدل بشفرة المسجل المراد التعامل معه .

الصورة العامة للأمر DCR هى :

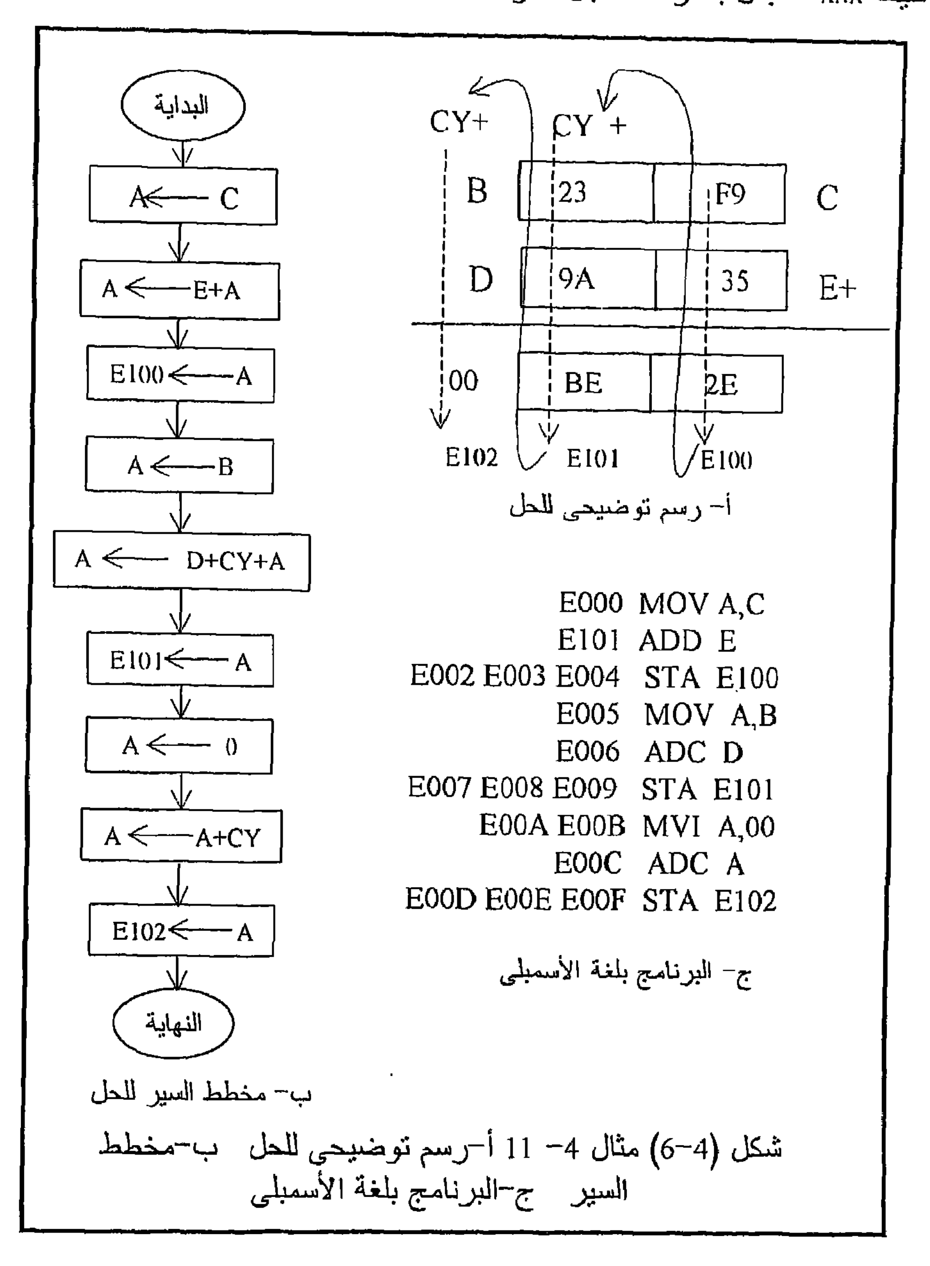
DCR reg

reg ⟵ reg − 1

حيث يقوم هذا الأمر بطرح واحد من محتويات المسجل reg أو مكان الذاكـرة M الذى عنوانه فى HL . الصورة الثنائية لهذا الأمر هى :

00xxx101

حيث xxx تستبدل بشفرة المسجل المراد التعامل معه .



شكل (4-6) مثال 4- 11 أ-رسم توضيحى للحل ب-مخطط السير ج-البرنامج بلغة الأسمبلى

4-4-5 الأمران INX و DCX

الأمران INX و DCX هما مرادفان للأمرين INR و DCR ولكنهما يستخدمان مع أزواج مسجلات ، فالأمر INX يجمع واحدا على محتويات زوج من المسجلات والصورة العامة الأسمبلى والثنائية له هى :

INX rp
00xx0011
rp⟵ 1 + rp

حيث xx تستبدل بشفرة زوج المسجلات المراد التعامل معه . وأما الأمر DCX فإنه يطرح واحدا من محتويات زوج من المسجلات وصورته العامة (الأسمبلى والثنائية) هى :

DCX rp
00xx1011
rp⟵ rp - 1

حيث تستبدل xx بشفرة زوج المسجلات المراد التعامل معه .

مثال 4-12

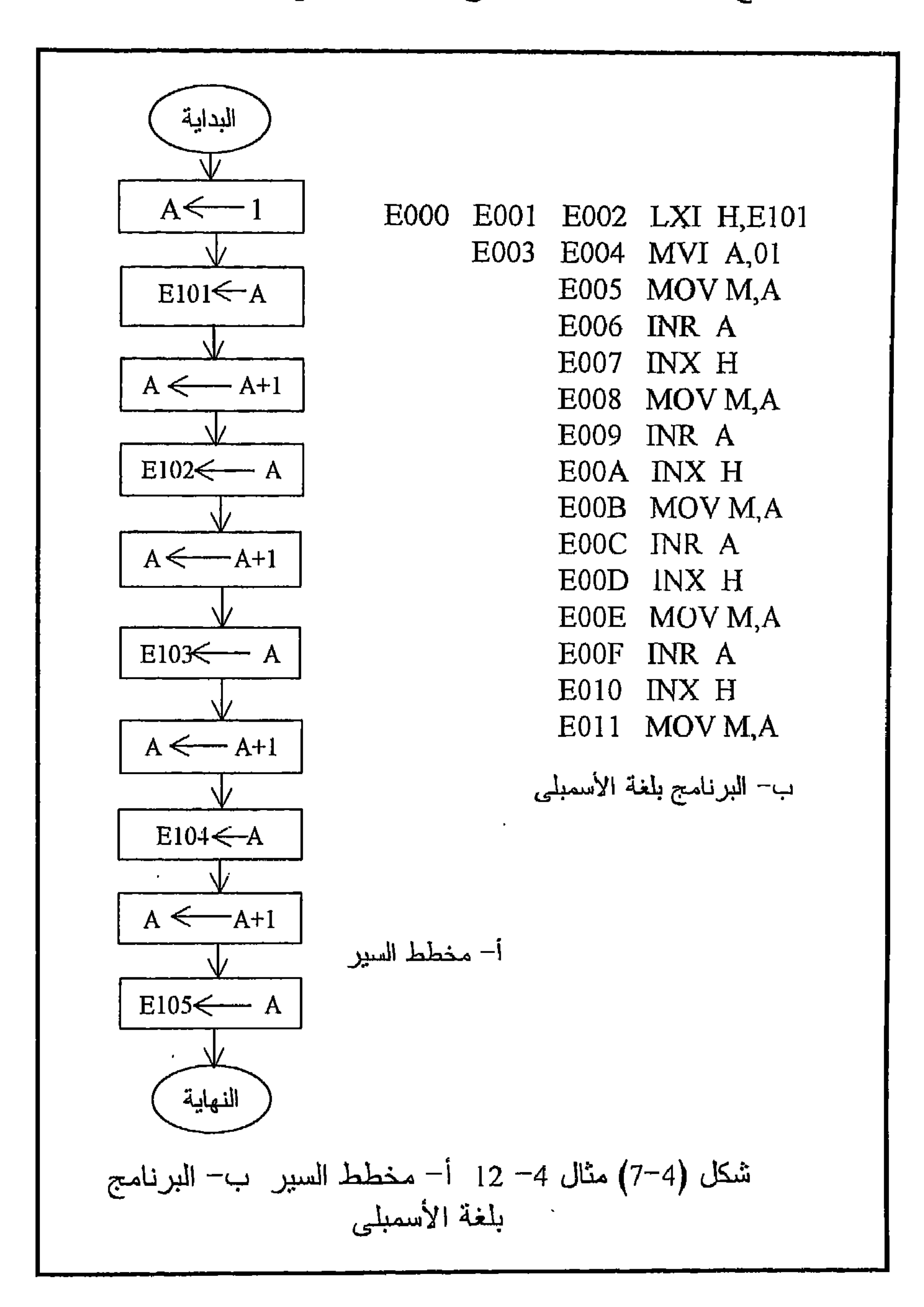
المطلوب تخزين الأرقام 01 ,02 ,03 ,04 ,05 فى أماكن الذاكرة التى عناوينها E101 ,E102 ,E103 ,E104 ,E105 على التوالى . شكل (4-7) يبين خريطة التدفق والبرنامج لحل هذه المسألة . لاحظ استخدام الأمر INR لزيادة واحد على محتويات المسجل A والأمر INX لزيادة واحد على محتويات زوج المسجلات HL فى هذا المثال . نلاحظ من شكل (4-7) أننا لكى نخزن خمسة أرقام فى الذاكرة كتبنا برنامجا مكونا من ثمانية عشرة بايت (E000 حتى E011) فما العمل لو أننا نريد تخزين ألف رقم أو أكثر ، كم ستشغل من ذاكرة للبرنامج..! هل هناك من حل لتخفيض عدد أوامر مثل هذه البرامج ؟ إن الحل لذلك هو استخدام أوامر القفز والحلقات وهو موضوع الجزء القادم بعد أن نعرض بعض التمارين كتطبيق على أوامر الحساب .

4-5 تمارين

1. اكتب برنامجا يجمع محتويات أماكن الذاكرة E101, E102, E103, E104, E105 مع ما يناظرها من الأماكن E10A, E10B, E10C, E10D, E10E ثم يضع النتيجة فى الأماكن .E110, E111, E112, E113, E114 .
2. أعد البرنامج السابق مستخدما الطرح بدلا من الجمع .

3. اكتب برنامجا يجمع الرقم F3A56BH مع الرقم 78B6A9H ويضع النتيجــــة في بايتات الذاكرة E103, E102, E101, E100 .

4. أعد البرنامج السابق مستخدما الطرح بدلا من الجمع .




```
E000  E001  E002  LXI  H,E101
      E003  E004  MVI  A,01
            E005  MOV  M,A
            E006  INR  A
            E007  INX  H
            E008  MOV  M,A
            E009  INR  A
            E00A  INX  H
            E00B  MOV  M,A
            E00C  INR  A
            E00D  INX  H
            E00E  MOV  M,A
            E00F  INR  A
            E010  INX  H
            E011  MOV  M,A
```

ب- البرنامج بلغة الأسمبلى

شكل (4-7) مثال 4- 12 أ- مخطط السير ب- البرنامج بلغة الأسمبلى

4-6 مجموعة أوامر القفز
Jump Instructions

القاعدة العامة أن المعالج يقوم بتنفيذ البرنامج حسب ترتيب الأوامر الموجودة فيه من أول البرنامج إلى نهايته ، ولقد كنا حريصين فى جميع الأمثلة السابقة على الحفاظ على هذه القاعدة ، ولكن هناك بعض التطبيقات التى تتطلب الخروج على هذه القاعدة كأن يطلب منك مثلا تنفيذ عملية معينة عدد معين من المرات أو عدد لا نهائى من المرات . فعندما يكون المعالج مثلا مراقبا لدرجة الحرارة فى عملية صناعية معينة فإن عليه أن يقرأ درجة الحرارة ويقارنها بدرجة حرارة مخزنة فى الذاكرة كمرجع ، وإذا زادت الحرارة عن حد معين يقوم المعالج بضرب جرس إنذار مثلا وإذا نقصت عن حد معين يشغل سخان لزيادتها ، مثل هذا البرنامج سيكون عبارة عن مجموعة من الأوامر التى تنفذ إلى مالانهاية طالما أن المعالج يراقب درجة الحرارة .

لقد أتاح المعالج هذه العملية بتوفير بعض الأوامر التى تمكنك كمبرمج من القفز بعملية التنفيذ من مكان لآخر خلال البرنامج . هناك نوعان من القفز :

4-6-1 القفز غير المشروط Unconditional jump

عند تنفيذ أى عملية قفز غير مشروط ينتقل المعالج بعملية التنفيذ إلى المكان الجديد دون أى قيد أو شرط ، وهناك أمر واحد فقط من أوامر المعالج 8085 يقوم بهذه العملية ، والصورة العامة لهذا الأمر هى :

JMP addr

حيث إنه عند تنفيذ هذا الأمر يوضع العنوان addr الذى سيتم القفز إليه فى عداد البرنامج فيصبح الأمر الموجود عند هذا العنوان هو الأمر الذى عليه الدور فى التنفيذ . لاحظ أن هذا الأمر يتكون من ثلاث بايتات واحدة هى شفرة الأمر واثنتان للعنوان addr الذى سيتم القفز إليه .

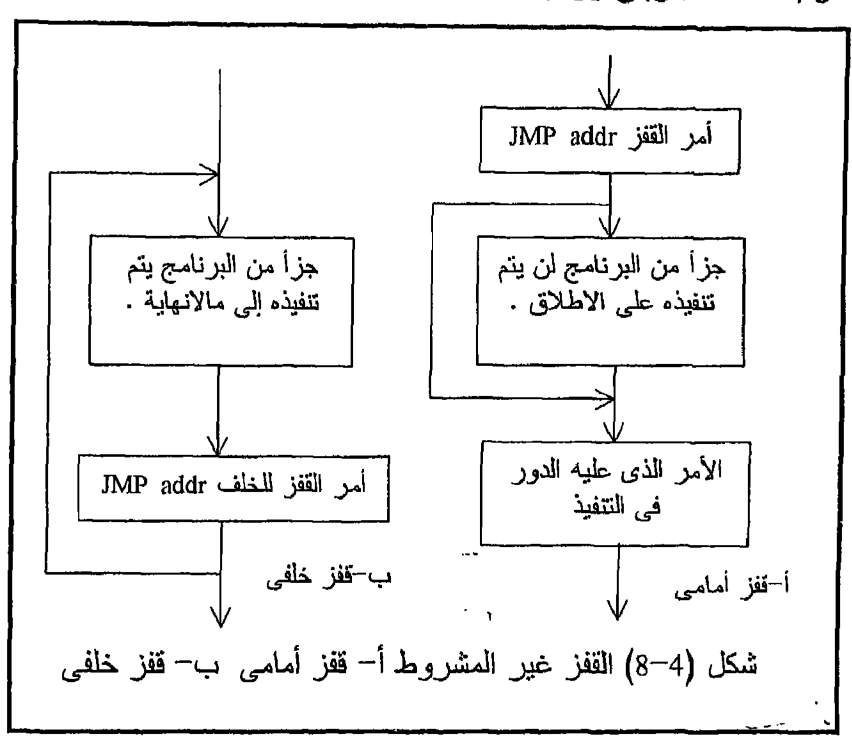
إن القفز باستخدام الأمر JMP addr قد يكون إلى الأمام فى البرنامج وقد يكون إلى الخلف . إذا كان القفز إلى الأمام سينتج عن ذلك وجود جزء من البرنامج لن ينفذ على الإطلاق وهو الجزء الذى بين أمر القفز JMP والأمر الذى سيتم القفز إليه . أما إذا كان القفز إلى الخلف فإنه سينتج عن ذلك ما يسمى بالحلقة اللانهائية Infinite loop والتى سوف يستمر المعالج فى تنفيذها إلى مالانهاية . شكل (4-8) يبين مخطط السير لعملية القفز غير المشروط الأمامى والخلفى .

4-6-2 القفز المشروط Conditional jump

كما يوحى الإسم فإنه فى هذا النوع من القفز لن يتم القفز إلا إذا تحقق شرط معين أما إذا لم يتحقق الشرط فإن البرنامج ينفذ فى التتابع الطبيعى حيث سينفذ

الأمر الذى بعد أمر القفز مباشرة . إن شروط القفز توضع دائما علــى الأعــلام التى فى مسجل الحالة ، فيمكنك مثلا أن تجعل القفز مشروطا بأن تكون النتيجــة صفرا أو تجعل القفز مشروطا بأن تكون النتيجة سالبة ، وهكذا ، حيث أن هنــاك خمسة أعلام واحد منها وهو علم الحمل النصفى HC لا يســتخدم كشــرط فــى عمليات القفز فإنه يتبقى أربعة أعلام يمكن أن تستخدم فى ثمانية من أوامر القفــز المشروط كالتالى :

اقفز إذا كانت النتيجة صفرا	JZ addr
اقفز إذا كانت النتيجة ليست صفرا	JNZ addr
اقفز إذا كانت النتيجة سالبة	JM addr
اقفز إذا كانت النتيجة موجبة	JP addr
اقفز إذا كان هناك حمل	JC addr
اقفز إذا لم يكن هناك حمل	JNC addr
اقفز إذا كانت الباريتى فردية	JPO addr
اقفز إذا كانت الباريتى زوجية	JPE addr



شكل (4-8) القفز غير المشروط أ- قفز أمامى ب- قفز خلفى

لاحظ أن عدد هذه الأوامر ثمانية ، إثنان منها لكل علم من الأعلام الأربعة تمثــل جميع الحالات التى يمكن أن يكون فيها هذا العلم (صفرا أو واحدا) . إن جميــع

هذه الأوامر لابد وأن تكون ثلاث بايتات ، واحدة هـــى شــفرة الأمـــر op code واثنتان للعنوان الذى سيتم القفز إليه إذا تحقق الشرط . لاحظ أن النتيجـــة التـــى نعنيها فى الأوامر السابقة هى آخر نتيجة تأثرت بها الأعلام ، ولذلك فإنه قبــل أن نكتب أى أمر من أوامر القفز غير المشروط يجب أن ندرس جيدا هـــل الأمـــر السابق لأمر القفز المشروط يؤثر على الأعلام أم لا كما ســيتضح مـــن المثـــال التالى:

<u>مثال 4-13</u>

اكتب برنامجا يقرأ محتويات بايت الذاكرة E100 باستمرار (إلى مالانهايـــة) ثـــم يختبر هذه المحتويات بحيث إذا كانت صفرا يضع 1 فى المسجل B وإذا كـــانت سالبة يضع 2 فى المسجل B وإذا كانت موجبة يضع 4 فى نفـــس المســـجل B . شكل (4-9) يبين خريطة التدفق والبرنامج لهذا المثال . البرنامج الموجـــود فـــى شكل (4-9) يعتبر تدريبا على معظم أفكار الحلقات ، ففيه الحلقـــات اللانهائيـــة الناتجة عن الأمر JMP addr ، كما أن فيه القفز المشروط JNZ addr . JP addr كما أن فيه القفز إلى الأمام فى البرنامج ، والقفز إلى الخلف أيضا. لذلـــك يجـــب دراسة هذا البرنامج بدقة وإمعان .

هناك أوامر أخرى يتسبب عنها قفز بعملية تنفيذ البرنامج إلى أماكن أخرى وهـــى أوامر القفز إلى البرامج الفرعية والعودة منها وسوف نرجىء الدراسة التفصيليـــة للبرامج الفرعية إلى فصل آخر خاص بذلك .

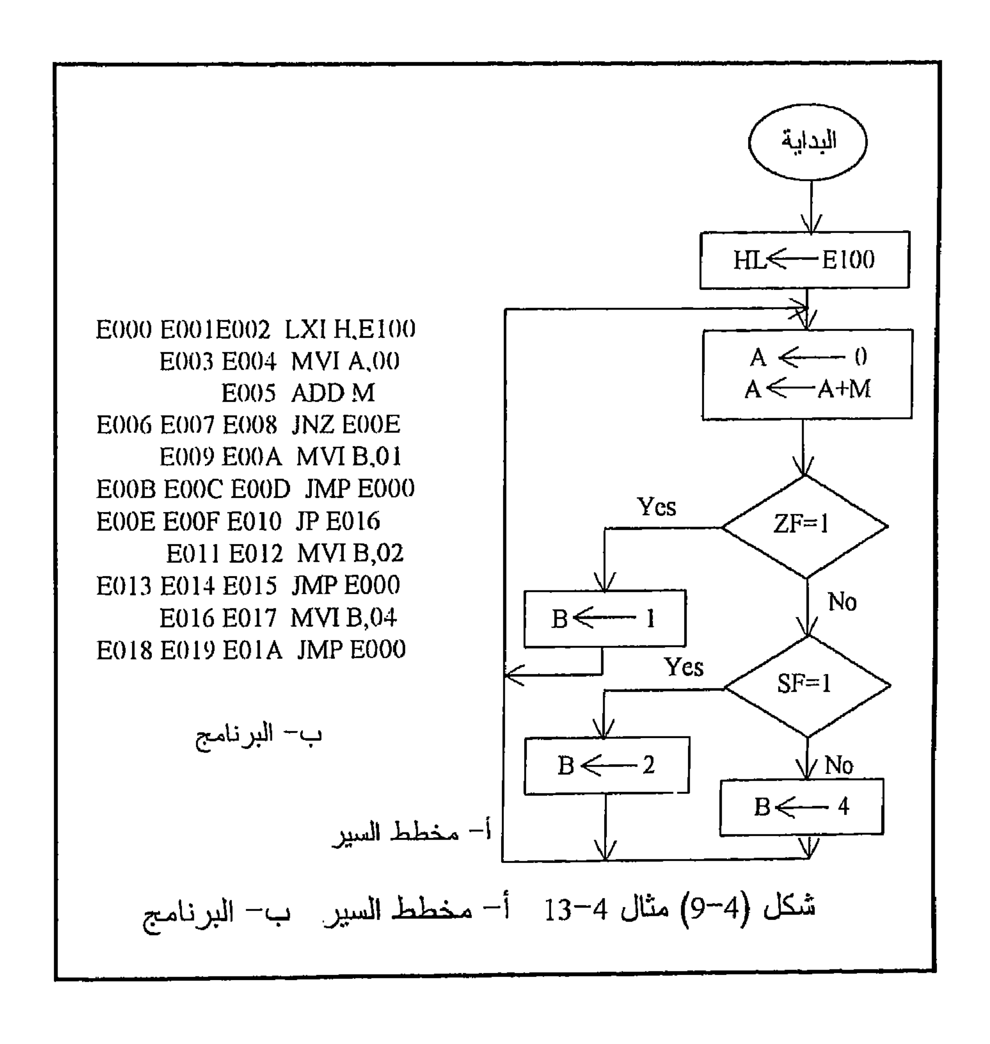
4-7 مهمة أخرى للأسمبلر

ألمهمة الوحيدة التى عرفناها حتى الآن للأسمبلر هى مهمة تحويـــل الشـــفرات الحرفية (الأسمبلى) إلى شفرات ثنائية أو لغة الماكينة ، ولكن لحسن الحـــظ فـــإن الأسمبلر قد تم تزويده ببعض المهام الأخرى التى تريح المبرمج إلى درجة كبيرة والتى سنتعرف على واحدة منها فى هذا الجزء . ينظر أى أسمبلر إلى الأمر الذى تكتبه له على أنه مكون من أجزاء أربعة كالموضحة فى شـــكل (4-10) والتـــى سندرسها بالترتيب فيما يلى :

<u>4-7-1 الأمر والمعاملات</u>

لقد درسنا الكثير من أوامر لغة الأسمبلى والتى سنلخصها كلها فى شـــكل واحـــد وفى ترتيب أبجدى فى آخر هذا الفصل إن شاء الله . من هذه الأوامـــر ,MOV ,ADD, JMP وغيرها . كل أمر من هذه الأوامر لابد وأن يكون لـــه معـــاملات وهذه المعاملات هى المسجلات أو أماكن الذاكرة التى سوف يؤثر عليها الأمـــر ،

فمثلا الأمر MOV A,B معاملاته هى المسجلين A, B والأمر ADD C معاملاتـــه هى المسجل C والأمر ADD M معاملاته هى بايت الذاكرة التى عنوانـــها فـــى المسجلين HL .... وهكذا . هناك أوامر قليلة ليست لها معاملات على الإطـــلاق ومنها الأمر NOP والذى معناه لا تعمل شـــيئا No Operation كمـــا أن عـــدد المعاملات فى أى أمر لا يزيد عن اثنين بأى حال . إن الأمر يجب أن يفصل عن معاملاته بمسافة واحدة على الأقل وإن زاد عدد المسافات عن واحدة فلن يضـــر ذلك فى شىء حيث أن الأسمبلر يهملها ، أما إذا لم توجد مسافة واحدة على الأقل بين الأمر ومعاملاته فإن الأسمبلر سيعطى خطأ على ذلك ولن يقبل منـــك هـــذا الأمر . إذا كان الأمر له معاملان كالأمر MOV A,B فإن المعـــاملين يجـــب أن يفصل بينهما بفاصلة (,) وإذا لم يحدث ذلك فإن الأسمبلر يعطى رسالة خطأ علـــى ذلك .



شكل (4-9) مثال 4-13 أ- مخطط السير ب- البرنامج

Label : علامة :	Command الأمر	Operand المعاملات	; Comment ; التعليق

شكل (4-10) أجزاء أمر لغة التجميع

4-7-2 التعليق Comment

حتى تجعل برنامجك مفهوما ومن السهل قراءته وتتبعه بالنسبة للآخرين فإنه يجب عليك أن تكتب بعض التعليقات البسيطة بجانب كل أمر . لذلك فإن الأسمبلر يعطيك فرصة أن تكتب أى شيء تريده فى نهاية الأمر على أن تفصل بين الأمر والتعليق بفاصلة منقوطة يفهم منها الأسمبلر أن كل المكتوب بعدها يعتبر تعليقا ولا يدخل ضمن الأمر . إذا كان التعليق الذى ستكتبه سيأخذ أكثر من خط واحد فإن كل خط يجب أن يبدأ بفاصلة منقوطة . المثال التالى سيوضح أهمية كتابة التعليق فى نهاية الأمر .

مثال 4. 14

```
                        ; Multiplication of two numbers
E000 E001  MVI A,00 ; Put 0 into A, Clean accumulator
E002  ADD B             ; Addition of A and B, result goes to A
E003 DCR  C             ; Decrement C by 1
E004 E005 E006   JNZ E002   ;Go to add B to itself one more time
                            ;as long as C not equal to 0
E007 E008 E009  STA E100   ; store result in location E100
;----------------------------------------------------------------------------
```

هذا البرنامج يضرب الرقم الموجود فى المسجل B فى الرقم الموجود فى المسجل C ويضع النتيجة فى بايت الذاكرة رقم E100 . بما أنه ليس هناك أمر من أوامر الشريحة 8085 يقوم بإجراء عملية الضرب لذلك كان لابد أن نكتب هذا البرنامج لإجراء عملية الضرب عن طريق تكرار الجمع حيث يجمع البرنامج محتويات المسجل B مع نفسه عدد من المرات يساوى الرقم الموجود فى المسجل C . ستجد أن التعليقات المكتوبة بجانب الأوامر تدل على ذلك . التعليق الموجود فى السطر الأول مثلا يقول "ضع صفرا فى A ، تنظيف المركم " وهذا ضرورى حتى لا نجمع أى رقم قد يكون فى المسجل A قبل البدء فى عملية جمع المسجل B مع نفسه . التعليق فى السطر الثانى يقول "جمع A+B ، النتيجة فى A " لاحظ

أن هذا الأمر هو بداية حلقة يتم القفز إليها من السطر الرابع وفى المرة الأولى من دخول هذه الحلقة سيجمع المعالج المسجل A=0 مع محتويات المسجل B فيصبح A=B . التعليق فى السطر الثالث يقول "انقص محتويات C بمقدار واحد". التعليق فى السطر الرابع يقول "ارجع لجمع المسجل B مع نفسه مرة أخرى طالما أن C لا يساوى صفرا ، النتيجة فى A ." التعليق فى السطر الخامس يقول " خزن النتيجة فى البايت E100" . التعليق فى السطر السادس وهو الخط المنقط تم وضعه فقط للدلالة على نهاية البرنامج وهذا أحيانا يكون مهما جدا فى عملية تنظيم كتابة البرنامج خاصة إذا كان البرنامج مكونا من أكثر من جزء حيث يمكن الفصل بين الأجزاء المختلفة عن طريق مثل هذه الخطوط التى يعرفها الأسمبلر على أنها تعليقات . بالطبع يجب أن تكون التعليقات باللغة الإنجليزية ، إلا إذا كان محرر الكلمات الذى تستخدمه فى كتابة البرنامج سيسمح لك باستخدام اللغة العربية فى التعليقات .

4-7-3 العلامة Label

فى جميع البرامج التى كتبناها حتى الآن كنا حريصين تماما على أن نكتب عناوين البايتات التى يشغلها أى أمر من أوامر البرنامج ، وكانت هذه العملية ضرورية بالذات عند التعامل مع أوامر القفز التى نحتاج فيها لمعرفة عنوان الأمر الذى سنقفز إليه مثل الأمر الرابع فى المثال السابق (مثال ضرب الرقمين 4-14) وهو JNZ E002 ، فلولا أننا كنا نكتب عناوين البايتات التى عندها الأوامر لما حددنا أن القفز يجب أن يكون إلى الأمر الثانى وهو E002 ADD B . فى الحقيقة إن مهمة تتبع العناوين لجميع أوامر البرنامج تعتبر مهمة صعبة للمبرمج وبالذات إذا كان البرنامج يحتوى على الكثير من أوامر القفز ، وتعتبر مهمة سهلة للأسمبلر يستطيع القيام بها فى نفس الوقت . فطالما أن الأسمبلر يعرف جيدا عدد البايتات التى يتكون منها كل أمر فلم لا يتولى هو عملية العنونة لأوامر البرنامج على أن يعطيه المبرمج فقط عنوان أول أمر فى البرنامج ، وما على المبرمج فى هذه الحالة إلا الكتابة فقط بالشفرات الأسمبلى ، هنا يبرز سؤال مهم : كيف سنحدد للأسمبلر الأوامر التى من الممكن أن يتم القفز إليها ؟ إن ذلك يتم عن طريق العلامات Labels التى نضعها قبل الأمر المراد القفز إليه على أن نستخدم نفس العلامة فى أمر القفز نفسه ، إننا سنعيد كتابة البرنامج الموجود فى المثال 4-14 مرة أخرى دون استخدام عناوين للأوامر وباستخدام العلامات كما فى المثال 4-15 .

مثال 4-15

```
MVI A,00     ; Put 0 into accumulator
```

```
HERE: ADD B      ; A+B into A
      DCR C      ; Decrement C by 1
      JNZ HERE   ; Jump to add B to itself one more
                 ; time as long as C not equal to 0
      STA E100   ; Store result in E100
;------------------------------------------------------------------------
```

لاحظ أننا وضعنا العلامة وهى كلمة :HERE كعلامة عند الأمر الذى سيتم القفز إليه وهو الأمر الثانى فى البرنامج السابق ، واستخدمنا نفس العلامة HERE فى الأمر JNZ HERE كى يتم القفز إلى الأمر رقم 2 . من البديهى أنه لابد وأن يكون هناك تطابق تام بين العلامة فى الأمر الذى سيتم القفز إليه والعلامة فى أمر القفز نفسه بحيث إذا لم يوجد هذا التطابق فإن الأسمبلر سيعطى خطأ . أيضا لابد وأن تنتهى العلامة الموجودة فى أول الأمر الذى سيتم القفز إليه بالحرف (:) ليميز بها الأسمبلر بين نهاية العلامة وبداية الأمر. كما أن عدد أحرف العلامة يجب ألا يزيد عن ثمانية أحرف لا تبدأ برقم. .

البرنامج الموجود فى المثال 4-14 مكتوب فى الذاكرة ابتداء من البايت E000 ، افترض مثلا أننا لأى سبب نريد نقل البرنامج بحيث يبدأ من البايت E050 بدلا من E000 . فى هذه الحالة لابد وأن نعيد فحص البرنامج بدقة ونغير العناوين الموجودة فى جميع أوامر القفز فى البرنامج ، فمثلا الأمر JNZ E002 فى المثال 4-14 سيصبح JNZ E052 لو أننا بدأنا البرنامج من البايت E050 ، وهكذا تخيل لو أن البرنامج به العديد من أوامر القفز فإنه لا شك ستكون عملية إعادة عنونة الأوامر من الصعوبة بمكان وبالذات فى البرامج الطويلة . إن استخدام العلامات Labels كما فى مثال 4-15 يريح المبرمج تماما من عملية تتبع العناوين فى أوامر القفز حتى لو تغير مكان البرنامج لأنك تعطى الأسمبلر عنوان بداية البرنامج فقط وهو ينسب جميع العلامات إلى عنوان البداية وهذه فى الحقيقة فائدة عظيمة يوفرها الأسمبلر للمبرمج . لذلك فإن جميع البرامج التى سنعرضها فيما بعد سنكتبها باستخدام لغة الأسمبلى والعلامات دون الحرص على كتابة عناوين الأوامر لأننا بذلك نكون قد عبرنا مرحلة لا بأس بها فى لغة الأسمبلى .

4-8 أوامر الإدخال والإخراج
Input Output instructions

إلى الآن رأينا كيف نبرمج شريحة المعالج وكيف نحرك المعلومات داخلها من مسجل إلى مسجل آخر أو من مسجل إلى الذاكرة أو العكس ولكن لم نعرف حتى الآن كيف نظهر معلومة على شاشة عرض مثلا أيا كان نوع هذه الشاشة ، أو

كيف ندخل معلومة إلى المعالج من خلال لوحة مفاتيح على سبيل المثال . إن شاشة العرض ولوحة المفاتيح يعتبران مثالان من ضمن العديد من الأمثلة التى تحتاج لعمليات الإخراج والإدخال . فمثلا حينما يستخدم المعالج فى التحكم فى أى متغير فى أى عملية صناعية وليكن مثلا درجة الحرارة فإنه لابد من إدخال درجة الحرارة إلى المعالج بعد تهيئتها ووضعها فى الصورة المناسبة لذلك ، وكذلك إذا أراد المعالج رفع درجة الحرارة أو ضرب جرس إنذار فإنه لابد وأن يخرج إشارة معينة تؤخذ وتهيأ فى الصورة المناسبة للجهاز الذى ستذهب إليه سواء كان سخانا أو جرسا أو غيره . جميع عمليات الإدخال والإخراج تتم من خلال ما يسمى ببوابات الإدخال والإخراج والتعامل مع هذه البوابات دائما ينقسم إلى قسمين : قسم خاص بالبناء الإلكترونى لهذه البوابات وكيفية توصيلها مع المعالج وهذا القسم سندرسه بالتفصيل فى فصل قادم إن شاء الله ، والقسم الآخر هو كيفية برمجة المعالج للتعامل مع هذه البوابات وهو موضوع دراستنا فى هذا الجزء حيث سندرس الأوامر الخاصة بهذه العملية وسنفترض فى دراستنا فى هذا الجزء أن القارىء لديه على الأقل بوابة إدخال واحدة وبوابة إخراج واحدة موصلان على الميكروكومبيوتر الذى يتدرب عليه ويكتب عليه برامجه .

4-8-1 أمر الإدخال IN وأمر الإخراج OUT

وهما اختصار لكلمتى Input يعنى أدخل و Output يعنى أخرج والصورة العامة للأمر IN هى :

IN no.

البوابة رقم .no ⟶ المسجل A

حيث سيقوم هذا الأمر بإدخال المعلومة الموجودة على بوابة الإدخال والتى رقمها .no فى الأمر نفسه إلى مسجل التراكم A . هذا الأمر يتكون من اثنين بايت واحدة هى شفرة الأمر IN والثانية هى رقم البوابة المراد التعامل معها ، ولذلك فإنه يمكن التعامل مع 2^8 = 256 بوابة إدخال تبدأ من البوابة رقم 00H إلى البوابة رقم FFH . انظر شفرة الأمر IN الستعشرية فى جدول الأوامر فى نهاية هذا الفصل.

الصورة العامة لأمر الإخراج OUT هى :

OUT no.

المسجل A ⟶ البوابة رقم .no

حيث سيقوم هذا الأمر بإخراج المعلومة الموجودة فى مسجل التراكم A إلى بوابة الإخراج التى رقمها .no . هذا الأمر يتكون من اثنين بايت واحدة هى شفرة الأمر OUT والثانية هى رقم البوابة المراد التعامل معها ، ولذلك فإنه يمكن التعامل مع 2^8 = 256 بوابة إخراج تبدأ أيضا من البوابة رقم 00H إلى البوابة رقم FFH. انظر شفرة الأمر OUT الستعشرية فى جدول الأوامر فى نهاية هذا الفصل .

لاحظ أن كلا من بوابة الإدخال وبوابة الإخراج تتكون من 8 بتات مثلها فى ذلك مثل أى مسجل يتكون من 8 بتات ولذلك فإن أكبر رقم يمكن أن يكون على أى بوابة هو الرقم 255 . تذكر أيضا أن أى تعامل مع أى بوابة باستخدام هذين الأمرين لابد وأن يكون من خلال المركم A ، فإذا أردت أن تخرج محتويات المسجل C مثلا إلى بوابة الإخراج رقم 05 فإنه لابد وأن تنقل هذه المحتويات إلى المسجل A أولا ثم تنفذ أمر الإخراج كما يلى :

MOV A,C ; انقل محتويات C إلى A

OUT 05 ; أخرج على بوابة الإخراج رقم 05

بعد تنفيذ الأمرين السابقين سترى محتويات المسجل C على بوابة الإخراج رقم 05 . أما إذا أردنا أن ندخل محتويات بوابة الإدخال رقم 00 ونخزنها فى بايت الذاكرة رقم E100 فإننا ننفذ الأمرين التاليين على سبيل المثال :

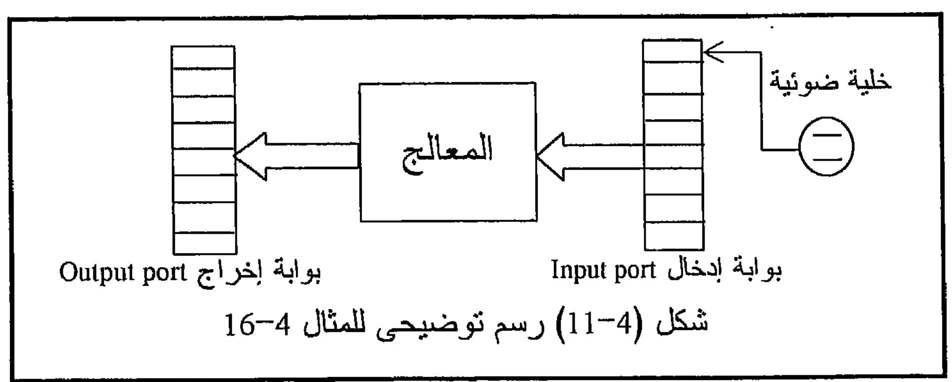
IN 00 ; أدخل محتويات بوابة الإدخال رقم 00 إلىالمركم

STA E100 ; خزن محتويات المركم فى البايت E100

بعد تنفيذ هذين الأمرين فإنه إذا كانت محتويات بوابة الإدخال رقم 00 تساوى 55 مثلا فإن هذا الرقم (55) ستجده فى البايت E100 .

<u>مثال 4-16</u>

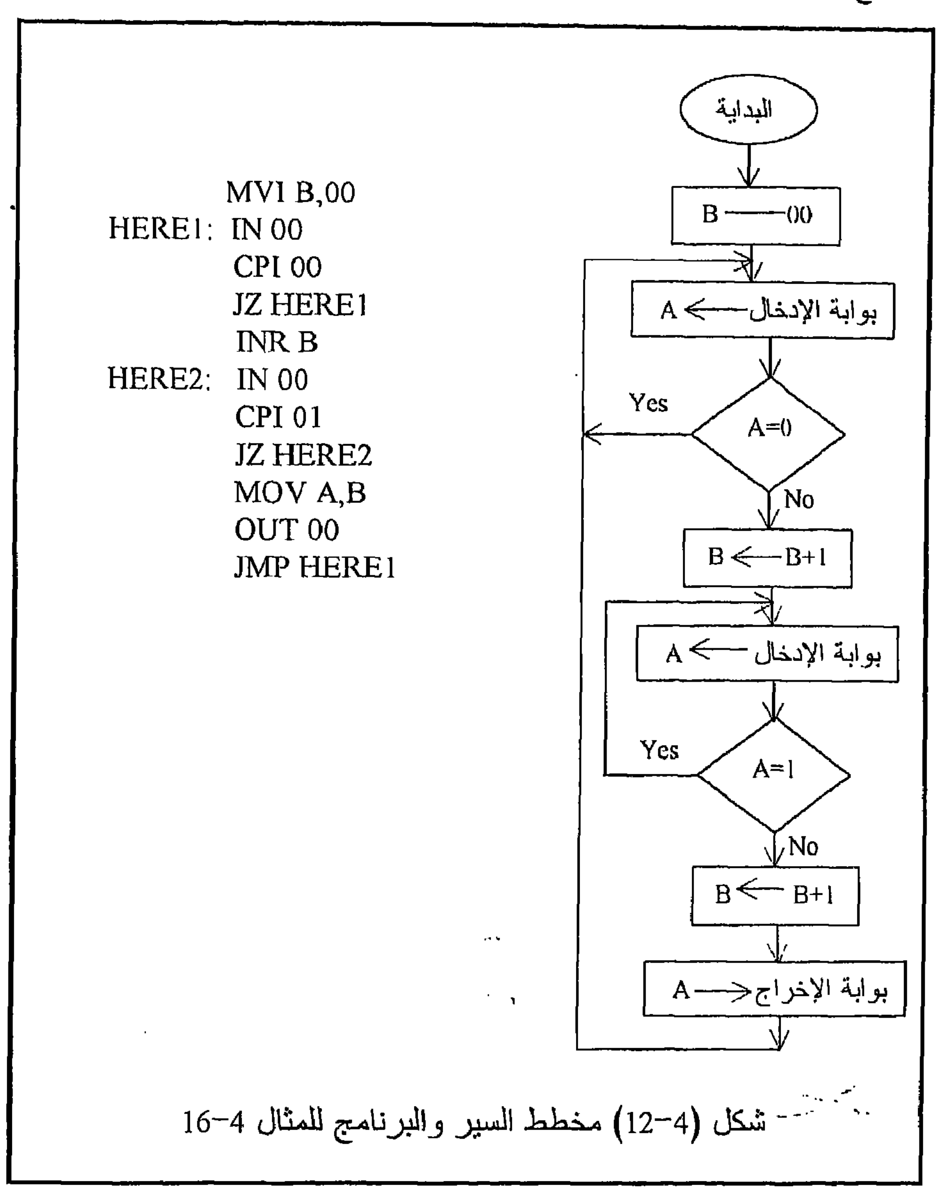
إفترض أن لدينا خط إنتاج فى أحد المصانع تعبر عليه المنتجات وفى أثناء العبور فإن كل منتج يقطع خلية ضوئية فتعطى نبضة كهربية على خرجها . خرج هذه الخلية موصل على البت رقم 0 فى بوابة الإدخال رقم 00H . المطلوب هو كتابة برنامج يعد هذه المنتجات ويخرج العدد على بوابة الإخراج رقم 00H أيضا . شكل (4-11) يبين رسما توضيحيا لهذه العملية وشكل (4-12) يبين مخطط السير والبرنامج لهذا المثال .



شكل (4-11) رسم توضيحى للمثال 4-16

لاحظ وجود الأمر CPI وهو أحد أوامر الحساب التى لم نشرحها وتركناها للقارىء لمراجعتها من قوائم الأوامر فى نهاية الفصل ، ولقد استخدم هذا الأمر

لمقارنة محتويات المسجل A القادمة من بوابة الإدخال بالقيمة 00 أولا ، وطالمـــا أن هذه المحتويات تساوى 00 فإن ذلك يعنى عدم مرور أى منتـــج علـــى خـــط الإنتاج ويقوم المعالج بتكرار عملية القراءة حيث يقفز إلـــى :HERE1 كمـــا فـــى البرنامج .



شكل (4-12) مخطط السير والبرنامج للمثال 4-16

عندما يمر أى منتج ستكون محتويات بوابة الإدخال مختلفة عن الصفر ولن يرجع المعالج إلى العلامة :HERE1 ثانية ولكنه سيزيد محتويات المســـجل B بمقـــدار

واحد حيث B تعتبر عدادا للمنتجات المارة على الخط ، ثم بعد ذلك سيقوم المعالج بقراءة بوابة الإدخال مرة أخرى للتأكد من أن المنتج قد مر من على الخط وذلك بمقارنة محتويات بوابة الإدخال بالقيمة 01 وطالما أنها تساوى 01 يقفز إلى العلامة :HERE2 حيث لا يعمل شيئا سوى قراءة البوابة . عندما تنزل البوابة إلى 00 مرة أخرى فإن ذلك يدل على أن المنتج قد مر على الخط فيقوم المعالج بإخراج قيمة العداد B على بوابة الإخراج ويذهب إلى العلامة :HERE1 مرة أخرى لمواصلة البرنامج .

4-9 مجموعة أوامر المنطق
Logic Instruction Set

العمليات المنطقية التى يستطيع المعالج 8085 القيام بها هى العمليات XOR, NOT, OR, AND وسنكتفى هنا بعرض الصورة العامة لهذه الأوامر على أن يقوم القارىء بمراجعة هذه الأوامر فى القوائم الموجودة فى آخر الفصل لمعرفة الشفرات الستعشرية والزمن الذى يأخذه كل أمر لكى يتم تنفيذه . كما ذكرنا سابقا فإن العمليات المنطقية مثلها مثل العمليات الحسابية لابد وأن يكون المركم طرفا فيها كما أن النتيجة لابد وأن توضع فى المركم أيضا . جميع العمليات المنطقية تؤثر على الأعلام ، انظر قوائم الأوامر فى آخر الفصل لترى ذلك ولترى متى تكون هذه الأوامر مكونة من بايت واحدة ومتى تكون مكونة من 2 بايت .

الصورة العامة للأمر AND هى :

ANA reg

A مسجل ⟵ A مسجل AND reg

حيث سيقوم المعالج بإجراء عملية AND على محتويات المسجل reg مع محتويات مسجل التراكم A ونتيجة العملية توضع فى المركم . بنفس الطريقة يمكننا كتابة الصورة العامة للأمرين OR و XOR كما يلى :

ORA reg

XRA reg

حيث سيقوم الأمر الأول بإجراء عملية OR على محتويات المسجل reg مع محتويات المسجل A ويضع النتيجة فى المسجل A . وأما الأمر الثانى فسيجرى عملية XOR على محتويات المسجل reg مع محتويات المركم ويضع النتيجة فى المركم . جميع العمليات المنطقية الثلاث السابقة يمكن أن تجرى على معلومة فورية أو ثابت وفى هذه الحالة فإن الصورة العامة لهذه الأوامر ستكون :

ANI data8

ORI data8

XRI data8

حيث data8 هو الثابت أو المعلومة الفورية المراد إجراء العملية المنطقية عليها .
لاحظ أن هذه الأوامر فى هذه الحالة سيكون كل منها مكونا من 2 بايت ، واحدة هى شفرة الأمر والثانية هى قيمة الثابت data8 .

مثال 4-17

افترض أن محتويات المسجل A تساوى FFH ومحتويات المسجل B تساوى F0H والمطلوب إجراء عمليات AND و OR و XOR على كل من المسجلين A و B .

أولا : عملية AND

A=11111111
B=11110000

النتيجة بعد إجراء الأمر ANA B هى : 11110000

أى أن محتويات المركم ستتغير إلى FOH . جميع الأعلام ستتأثر بهذه النتيجة ما عدا علمى الحمل والحمل النصفى حيث يكونان صفرا دائما فى جميع العمليات المنطقية لأنه لا يحدث لا حمل ولا حمل نصفى مع أى عملية منطقية .

ثانيا : عملية OR

A=11111111
B=11110000

النتيجة بعد إجراء الأمر ORA B هى : 11111111

أى أن محتويات المركم ستظل FFH وستتأثر الأعلام بنفس الطريقة التى ذكرناها مع عملية AND .

ثالثا : عملية XOR

A=11111111
B=11110000

النتيجة بعد إجراء الأمر XRA B هى : 00001111

أى أن محتويات المركم ستصبح 0FH وستتأثر جميع الأعلام بنفس الطريقة السابقة .

بذلك نكون قد أنهينا دراسة المجموعات الأساسية فى أوامر المعالج 8085 على أننا لم نذكر بالطبع جميع الأوامر داخل كل مجموعة ولكننا بذلك نكون قد عرفنا الغالبية العظمى من الأوامر وما تبقى منها فمن السهل معرفته من جداول الأوامر الموجودة فى آخر هذا الفصل كما أن هناك أبواب خاصة ستتعرض لمجموعات معينة من الأوامر مثل الأوامر الخاصة بالبرامج الفرعية والأوامر الخاصة بالمقاطعة حيث سيفرد فصل خاص لكل منها إن شاء الله نتيجة لأهميتها .

4-10 كيفية الإتصال بالذاكرة
Memory addressing

أى معلومة من حيث المكان إما أن تكون موجودة داخل المعالج نفسة فى أى مسجل من مسجلاته وفى هذه الحالة فإنه يمكن التعامل معها بسهولة وتحريكها من مكان لآخر داخل المعالج فى زمن أقل ، وإما أن تكون موجودة فى الذاكرة ويريد المعالج قراءتها ، أى إحضارها من الذاكرة ووضعها فى أى مسجل من مسجلاته ، أو كتابتها أى نقلها إلى الذاكرة ، وذلك يتطلب من المعالج أن يحدد عنوان المكان أو البايت فى الذاكرة التى ستتم معها عملية القراءة أو الكتابة . طريقة تحديد عنوان البايت من الذاكرة المراد التعامل معها هى المقصود دراسته فى هذا الجزء . يجب أن نتذكر جيدا أن جميع شرائح المعالجات التى نتعامل معها إلى الآن لها مسار عناوين به 16 بتا ولذلك فإننا عندما سنكتب أى عنوان فى النظام الستعشرى فلابد أن نكتبه من 4 خانات حيث كل خانة فى النظام الستعشرى تمثل 4 بتات فى النظام الثنائى ، فمثلا العنوان 1111000010100001 فى النظام الثنائى هو F0A1H فى النظام الستعشرى وغالبا نضع الحرف H للدلالة على أن الرقم مكتوبا فى النظام الستعشرى .
هناك طريقتان يحدد بهما المعالج 8085 عنوان المكان أو البايت فى الذاكرة المراد التعامل معه وسنبين هاتين الطريقتين فيما يلى :

4-10-1 الطريقة المباشرة Direct method

فى هذه الطريقة فإن الأمر نفسه يحتوى عنوان البايت المراد التعامل معها ، ولذلك فإن كل الأوامر التى تقع تحت هذا الصنف تتكون من ثلاث بايتات ، واحدة من هذه البايتات هى شفرة الأمر operation code واختصارا تكتب op code والإثنين بايت التالية هى عنوان المكان فى الذاكرة المراد التعامل معه . تذكر أن البايت تتكون من 8 بتات وأنه دائما تكون البايت الأولى من بايتات العنوان هى البايت ذات القيمة الصغرى Low significant byte . بالنسبة للمعالج 8085 هناك الأمران LDA addr و STA addr كأمثلة على الأوامر التى تتعامل بهذه الطريقة. كما رأينا سابقا فإن الأمر LDA addr يقوم بتحميل المسجل A (المركم) بمحتويات المكان فى الذاكرة الذى عنوانه موجود فى الأمر نفسه وقد رمزنا له بالرمز addr . أما الأمر STA addr فإنه يفعل العكس حيث يقوم بتخزين محتويات المسجل A فى المكان الذى عنوانه addr فى الذاكرة .

4-10-2 الطريقة غير المباشرة Indirect method

فى هذه الطريقة يوضع عنوان البايت المراد التعامل معها فى المسجلين H و L بحيث يحتوى المسجل H على البايت ذات القيمة العظمى من العنوان ويحتوى

المسجل L على البايت ذات القيمة الصغرى منه . مثل هذه الأوامر تتكون دائمـــا من بايت واحدة حيث يرمز لهذا المكان فى الذاكرة بالرمز M وتأخذ الشـفرة 110 كما رأينا من خلال تعاملنا مع الأوامر سابقا . فمثلا الأمـو MOV A,M معنـاه انقل محتويات بايت الذاكرة التى عنوانها فى زوج المسجلات HL إلـــى مســجل التراكم .

السؤال الذى يتبادر إلى الذهن هنا أى الطريقتين يفضل فى الاستخدام ، الطريقـــة المباشرة أم غير المباشرة ؟ الإجابة عن هذا السؤال تتوقف على البرنامج أو على المشكلة التى نبرمجها . فإذا كان البرنامج يتعامل مع الذاكرة باستمرار وبـــالذات إذا كان التعامل مع أماكن متعاقبة فى الذاكرة فإن الطريقة غير المباشرة لا شـــك تكون الأفضل لأنها ستوفر الكثير من طول البرنامج لأن المسجلين HL فى هـــذه الحالة سيكونان عبارة عن مؤشر إلى بايت الذاكرة التى تستخدمها وكلمـــا أردت التعامل مع بايت جديدة تزيد محتويات المسجلين HL بواحد كما رأينا فى المثـــال 4-7 . أما إذا كنت ستتعامل مع الذاكرة لمرة واحدة فإنـــه ليـس هنـــاك أى داع لاستخدام الطريقة غير المباشرة ولكن يفضل فى هذه الحالة الطريقة المباشرة .

الأشكال التالية تحتوى مجموعة أوامر المعالج 8085 مقسمة أولا إلى مجموعـــات كما أشرنا سابقا بحيث يحتوى كل شكل الأوامر الخاصة بمجموعة معينة والشفرة الستعشرية وعدد نبضات التزامن التى يحتاجها كل أمر لكى يتم إحضــــاره مـــن الذاكرة وتنفيذه . شكل (4-18) يبين قائمة أوامر الشريحة 8085 مرتبـــة ترتيبـــا أبجديا مع ملخص أو نبذة عن وظيفة كل أمر وكذلك الأعلام التى تتأثر بكل أمر .

MOV	A	B	C	D	E	H	L	M
,A	7F	47	4F	57	5F	67	6F	77
,B	78	40	48	50	58	60	68	70
,C	79	41	49	51	59	61	69	71
,D	7A	42	4A	52	5A	62	6A	72
,E	7B	43	4B	53	5B	63	6B	73
.H	7C	44	4C	54	5C	64	6C	74
.L	7D	45	4D	55	5D	65	6D	75
.M	7E	46	4E	56	5E	66	6E	

MVI	A	B	C	D	E	H	L	M
	3Exx	06xx	0Exx	16xx	1Exx	26xx	2Exx	36xx

LDA (13)	STA (13)
3Axxxx	32xxxx

LDAX B (7)	LDAX D (7)	STAX B (7)	STAX D (7)
0A	1A	02	12

IN (10)	OUT (10)	LHLD (16)	SHLD (16)
DBxx	D3xx	2Axxxx	22xxxx

XCHG (4)	XTHL (16)	SPHL (6)
EB	E3	F9

	LXI (10)	POP (10)	PUSH (10)
B	01xxxx	C1	C5
D	11xxxx	D1	D5
H	21xxxx	E1	E5
SP	31xxxx		
PSW		F1	F5

- جميع الأوامر التى على الصورة MOV M,reg و MOV reg,M تأخذ 7 نبضات تزامن والأوامر التى على الصورة MOV reg,reg تأخذ 4 نبضات تزامن .
- xx تعنى معلومة 8 بتات (data8) و xxxx تعنى عنوان أو معلومة 16 بتا
- جميع الأوامر التى على الصورة MVI reg,xx تأخذ 7 نبضات تزامن حتى يتم إحضارها وتنفيذها والأمر MVI M,xx يأخذ 10 نبضات.
- الأرقام التى بين القوسين للأوامر الأخرى تدل على عدد النبضات التى يأخذها الأمر.

شكل (4-13) مجموعة أوامر الانتقال للمعالج 8085

	INR	DCR	ADD	SUB	ADC	SBB
A	3C	3D	87	97	8F	9F
B	04	05	80	90	88	98
C	0C	0D	81	91	89	99
D	14	15	82	92	8A	9A
E	1C	1D	83	93	8B	9B
H	24	25	84	94	8C	9C
L	2C	2D	85	95	8D	9D
M	34	35	86	96	8E	9E

	INX	DCX	DAD
B	03	0B	09
D	13	1B	19
H	23	2B	29
SP	33	3B	39

ADI(7)	SUI(7)	ACI(7)	SBI(7)	DAA(4)
C6xx	D6xx	CExx	DExx	27

- الأوامر ADD M ، SUB M ، ADC M ، SBB M كلها تأخذ 7 نبضات لكى يتم إحضارها وتنفيذها .
- الأمران DCR M INR M تأخذ 10 نبضات .
- جميع أوامر الحساب الأخرى تحتاج إلى 4 نبضات .
- الأرقام التى بين القوسين لبعض الأوامر تدل على عدد النبضات اللازمة لإحضار وتنفيذ هذا الأمر.

شكل (4-14) مجموعة أوامر الحساب

	ANA	ORA	XRA	CMP
A	A7	B7	AF	BF
B	B0	A0	A8	B8
C	A1	B1	A9	B9
D	A2	B2	AA	BA
E	A3	B3	AB	BB
H	A4	B4	AC	BC
L	A5	B5	AD	BD
M	A6	B6	AE	BE

ANI	ORI	XRI	CPI	CMA
E6xx	F6xx	EExx	FExx	2F

- جميع أوامر المنطق تأخذ 4 نبضات ما عدا الأوامر التى تتعامل مع ذاكرة M والأوامر ANI ORI XRI CPI فتأخذ7 نبضات لكى يتم إحضارها وتنفيذها.

شكل (4-15) مجموعة أوامر المنطق

JMP	JC	JNC	JZ	JNZ	JM	JP	JPE	JPO
C3 XXXX	DA XXXX	D2 XXXX	CA XXXX	C2 XXXX	FA XXXX	F2 XXXX	EA XXXX	E2 XXXX

- هذه الأوامر إذا تم لها القفز تأخذ 10 نبضات وإذا لم يتم القفز تأخذ 7 نبضات لكى يتم إحضارها وتنفيذها .

CALL	CC	CNC	CZ	CNZ	CM	CP	CPE	CPO
CD XXXX	DC XXXX	D4 XXXX	CC XXXX	C4 XXXX	FC XXXX	F4 XXXX	EC XXXX	E4 XXXX

- هذه الأوامر تأخذ 18 نبضة إذا تم لها القفز وإذا لم يتم القفز تأخذ 9 نبضات .

RET	RC	RNC	RZ	RNZ	RM	RP	RPE	RPO
C9	D8	D0	C8	C0	F8	F0	E8	E0

- هذه الأوامر تأخذ 12 نبضة إذا تم القفز و 6 إذا لم يتم ما عدا الأمرRET فإنه يأخذ 10 نبضات .

RST 7	RST 6	RST 5	RST 4	RST 3	RST 2	RST 1	RST 0
FF	F7	EF	E7	DF	D7	CF	C7

- هذه الأوامر تأخذ 12 نبضة لكى يتم إحضارها وتنفيذها .

PCHL
E9

- هذا الأمر يحتاج إلى 6 نبضات لكى يتم إحضاره وتنفيذه .

شكل (4-16) مجموعة أوامر القفز

RLC	RRC	RAL	RAR	CMC	STC	EI	DI	HLT	NOP
07	0F	17	1F	3F	37	FB	F3	76	00

RIM	SIM
20	30

- جميع هذه الأوامر تحتاج إلى 4 نبضات لكى يتم إحضارها وتنفيذها .

شكل (4-17) أوامر أخرى

الأسمبلى	الشفرة	الأعلام المتأثرة	وظيفة الأمر
ACI const	11001110 data8	Z S P CY HC	A ← A + علم الحمل+ثابت
ADD reg	10000sss	Z S P CY HC	A ← A + مسجل
ADC reg	10001sss	Z S P CY HC	A ← A + علم الحمل+مسجل
ADI const	11000110 data8	Z S P CY HC	A ← A + ثابت
ANA reg	10100sss	Z S P 0 0	A ← A AND مسجل
ANI const	11100110 data8	Z S P 0 0	A ← A AND ثابت
CALL addr	CD addr16		نداء غير مشروط على برنامج فرعى
CC addr	DC addr16		نداء برنامج فرعى مشروط بعلم الحمل=1
CM addr	FC addr16		نداء برنامج فرعى مشروط بنتيجة سالبة
CMA	2F		اعكس محتويات المسجل A
CMC	3F		اعكس علم الحمل
reg CMP	10111sss	Z S P CY HC	قارن مسجل مع مسجلA. اطرح (reg – A) المسجل A لا يتأثر بالنتيجة.
CNC addr	D4 addr16		نداء برنامج فرعى مشروط بعلم الحمل= 0
CNZ addr	C4 addr16		نداء برنامج فرعى مشروط بعلم الصفر=0
CP addr	F4 addr16		نداء برنامج فرعى مشروط بنتيجة موجبة
CPE addr	EC addr16		نداء برنامج فرعى مشروط بباريتى زوجية
CPI const	FE data8	Z S P CY HC	مقارنة , (ثابت – A) والمسجل A لا يتاثر
CPO addr	E4 addr16		نداء برنامج فرعى مشروط بباريتى فردية
CZ addr	CC addr16		نداء برنامج فرعى مشروط بعلم الصفر=1
DAA	27	Z S P CY HC	حول المركم إلى النظام العشرى

DAD rp	00rp1001	CY	اجمع HL ← HL + rp
DCR reg	00ddd101	Z S P CY HC	انقص محتويات المسجل reg بمقدار 1
DCX rp	00rp1011		انقص محتويات المسجلين rp بمقدار 1
DI	F3		اهمل المقاطعة
EI	FB		اسمح بالمقاطعة
HLT	76		أوقف تنفيذ البرنامج
IN no.	DB Port no.		اقرأ بوابة الإدخال رقم no.
INR reg	00ddd100	Z S P CY HC	اجمع 1 على محتويات المسجل reg
INX rp	00rp0011		اجمع 1 على محتويات المسجلين rp
JC addr	DA addr16		قفز مشروط بعلم الحمل = 1
JM addr	FA addr16		قفز مشروط بعلم الإشارة = 1
JMP addr	C3 addr16		قفز غير مشروط
JNC addr	D2 addr16		قفز مشروط بعلم الحمل = 0
JNZ addr	C2 addr16		قفز مشروط بعلم الصفر = 0
JP addr	F2 addr16		قفز مشروط بعلم الإشارة = 0
JPE addr	EA addr16		قفز مشروط بباريتى زوجية
JPO addr	E2 addr16		قفز مشروط بباريتى فردية
JZ addr	CA addr16		قفز مشروط بعلم الصفر = 1
LDA addr	3A addr16		محتويات عنوان ← المسجل A
LDAX rp	00rp1010		محتويات العنوان الموجود فى الزوج BC أو DE تنقل للمسجل A
LHLD addr	2A addr16		محتويات العنوان addr والذى يليه ← المسجلين HL
LXI rp,const	00rp0001		ثابت من 16 بت ← زوج

	data16		المسجلات rp
MOVreg1,reg2	01dddsss		محتويات reg2 ⟵ reg1
MVIreg,const	00ddd110 data8		ثابت من 8 بت ⟵ reg
NOP	00		لا تعمل شيء
ORA reg	10110sss	Z S P 0 0	OR المسجل reg مع المسجل A
ORI const	F6 data8	Z S P 0 0	OR ثابت من 8 بت مع المسجل A
OUT no.	D3 Port no.		المسجل A ⟵ بوابة الإخراج رقم.no
PCHL	E9		قفز للعنوان الموجود فى المسجلين HL
POP rp	11rp0001		محتويات قمة المكدسة ⟵ rp
PUSH rp	11rp0101		محتويات rp ⟵ قمة المكدسة
RAL	17	CY	دوران المركم للشمال من خلال علم الحمل
RAR	1F	CY	دوران المركم لليمين من خلال علم الحمل
RC	D8		عودة مشروطة بعلم الحمل = 1
RET	C9		عودة غير مشروطة
RLC	07	CY	دوران المركم للشمال , آخر بت الى البت الأولى وإلى علم الحمل ولا يذهب علم الحمل لأول بت
RM	F8		عودة مشروطة بنتيجة سالبة
RNC	D0		عودة مشروطة بعلم الحمل = 0
RNZ	C0		عودة مشروطة بعلم الصفر = 0
RP	F0		عودة مشروطة بنتيجة موجبة
RPE	E8		عودة مشروطة بباريتى زوجية
RRC	0F	CY	دوران المركم لليمين , أول بت إلى أخر بت وإلى علم الحمل ولا يذهب علم الحمل لآخر بت
RST	11nnn111		ابدأ من العنوان صفر
RZ	C8		عودة مشروطة بعلم الصفر = 1
RIM	20		اقرأ قناع المقاطعة

SBB reg	10010sss	Z S P CY HC	طرح reg وعلم الحمل من المسجل A
SBI const	DE data8	Z S P CY HC	اطرح ثابت وعلم الحمل من المسجل A
SHLD	22		خزن محتويات المسجلين HL في الذاكرة
SIM	30		ضع قناع المقاطعة
SPHL	F9		حمل مؤشر المكدسة من المسجلين HL
STA add	32 addr16		خزن محتويات المركم في العنوان addr
STAX rp	00rp0010		خزن المركم في العنوان الموجود في المسجلين rp (BC و DE فقط)
STC	37	– – – 1 –	اجعل علم الحمل =1
SUB reg	10010sss	Z S P CY HC	اطرح المسجل reg من المسجلA
SUI const	D6 data8	Z S P CY HC	اطرح ثابت من المسجل A
XCHG	EB		بدل محتويات المسجلين DE مع HL
XRA reg	10101sss	Z S P 0 0	XOR المسجل reg مع المركم
XRI const	EE data8	Z S P 0 0	XOR ثابت مع المركم
XCHG	EB		بدل محتويات المسجلين HL مع مؤشر المكدسة

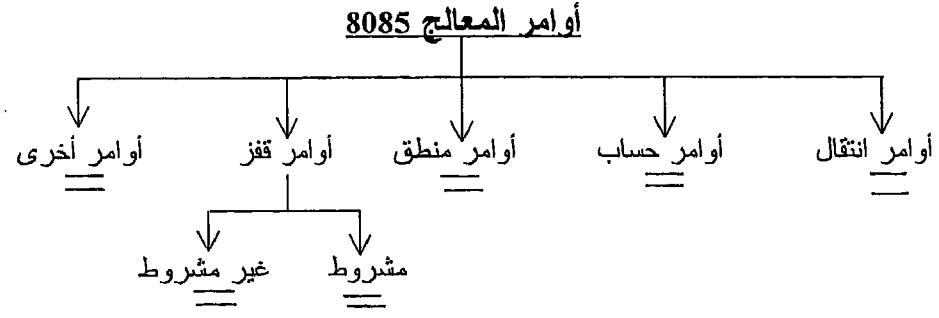
ملاحظات :

- **reg** اختصار لكلمة register وتعني مسجل 8 بت .
- **rp** اختصار لكلمة register pair وتعني زوجا من المسجلات .
- **const** اختصار لكلمة constant وتعني ثابت أو معلومة فورية وأحيانا يكون هذا الثابت 8 بتات وأحيانا يكون 16 بتا على حسب الأمر إذا كان يتعامل مع مسجل واحد أو زوج من المسجلات .
- **addr** اختصار لكلمة address وتعنيعنوانا في الذاكرة ودائما يكون العنوان 16بتا .
- **Z** تعني علم الصفر S
- تعني علم الإشارة
- **P** تعنيعلم الباريتي
- **CY** تعني علم الحمل HC
- تعني علم الحمل النصفي .

شكل (4. 18) أوامر الشريحة 8085 مرتبة أبجديا

11-4 تــــمـــــاريــــن

1. أكمل الجدول التالى الخاص بأوامر الشريحة 8085 :



2. ما هى نتيجة تنفيذ البرنامج التالى :

```
E000  MVI A,05
E002  MOV B,A
E003  MOV C,B
E004  MOV D,C
E005  MOV E,D
E006  MOV L,E
E007  MOV H,L
E008  MOV M,L
```

3. اقرأ البرنامج السابق وأجب عما يلى :

- محتويات مكان الذاكرة ....=E000
- محتويات مكان الذاكرة ....=E001
- محتويات مكان الذاكرة ....=0505

4.

```
E000 LXI H,E100
E003 MVI M,3E
E005 INX H
E006 MVI M,05
E008 INX H
E009 MVI M,47
E00B INX H
E00C MVI M,48
```

ما هى نتيجة تنفيذ البرنامج السابق ؟

5. على ضوء نتيجة تنفيذ البرنامج السابق ما هى نتيجة تنفيذ الشفرات الموجـودة فى الأماكن E100 إلى E103 ؟
6. هل تتأثر الأعلام بأوامر الانتقال ؟
7. اذكر الأعلام التى تتأثر بكل عملية من العمليات الحسابية والمنطقية ؟
8. إذا كانت محتويات المسجل A=F3H ومحتويات المسـجل B=A4H فـاكتب محتويات المسجل A بعد تنفيذ كل أمر من الأوامر التالية على نفس المحتويـات السابقة ووضح أيضا كيف ستتأثر الأعلام بكل أمر :

```
ADD B
SUB B
SUB A
INC A
ANA B
ORA B
XRA B
```

9. ارسم خريطة تدفق للبرنامج التالى وما هى نتيجة تنفيذه :

```
E000 MVI L,50H
E003 MVI H,E1H
E005 MOV M,A
E006 DCR L
E007 JNZ E005
```

10. ماذا يحدث لو كتبنا البرنامج السابق عند E100 بدلا من E000 ؟
11. أعد كتابة البرنامج السابق مستخدما العلامات Labels ؟ وما هـى مميـزات البرنامج مكتوبا بهذه الصورة ؟
12. كم عدد بوابات الإدخال التى يستطيع المعالج 8085 التعامل معها؟
13. كم عدد بوابات الإخراج التى يستطيع المعالج 8085 التعامل معها؟
14. على ماذا يتوقف هذا العدد ؟
15. هل هناك ما يمنع أن تكون بوابتى إدخال وإخراج لهما نفس الرقـم . كمثـال على ذلك IN 05 و OUT 05 ؟
16. OUT C,reg هذا أحد أوامر الإخراج للبروسيسور Z80 والذى يعنى إخـراج محتويات المسجل reg على بوابة الإخراج التى رقمها فى المسجل C فهل المعالج 8085 لديه ما يكافىء هذا الأمر ؟
17. هل تتأثر الأعلام بأوامر الإدخال والإخراج ؟
18. أكتب برنامجا يقرأ محتويات البوابة 00 وإذا كانت هذه المحتويـات زوجيـة يخزنها فى الذاكرة ابتداء من العنوان E100 وإذا كانت فرديـة يخرجـها علـى البوابة 00 ؟

19. المعالج 8085 لديه طريقتان فقط لعنونة الذاكرة وهما العنونة المباشرة والعنونة غير المباشرة ، اذكر الأوامر التى تستخدم مع كل من الطريقتين ؟
20. اذكر متى تفضل استخدام العنونة المباشرة ومتى تفضل العنونة غير المباشرة؟
21. المعالجان Z80 و MC6800 بهما طرق أخرى لعنونة الذاكرة والتى منها على سبيل المثال العنونة المفهرسة indexed addressing فهل هناك طرق أخرى لعنونة الذاكرة لدى المعالج 8085 ؟
22. اكتب برنامجا يحسب عدد الوحايد الموجودة فى محتويات المسجل A ، فمثلا إذا كان A=11110101 فإن عدد الوحايد = 6 .
23. اكتب برنامج يحسب أكبر قيمة عددية فى لبايت فى المدى العنوانى E200H إلى E250H .
24. اكتب برنامج يحسب عدد البايتات التى تحتوى أصفرا والتى تحتوى أرقاما موجبة والتى تحتوى أرقاما سالبة فى المدى العنوانى E100 إلى E150 .
25. اكتب برنامج يحسب عدد البايتات التى تحتوى بيانات فردية والتى تحتوى بيانات زوجية فى المدى العنوانى E100 إلى E150 .
26. المدى العنوانى E100 إلى E150 يحتوى بيانات لإشارة صوت ، احسب كم مرة عبرت إشارة الصوت الصفر .
27. اكتب برنامج يقرأ بوابة الإدخال رقم 00 ويختبر البت الرابعة فيها ، فإذا كانت هذه البت واحد يخرج هذه المحتويات على البوابة 00 ، وإذا كانت هذه البت صفر يخرج محتوياتها على البوابة 01 .
28. اكتب برنامج يقرأ بوابة الإدخال رقم 00 إلى مالانهاية ويختبر البيانات التى يقرأها ، فإن كانت فردية يخرجها على البوابة 00 ، وإن كانت زوجية يخرجها على البوابة 01 . احسب أكبر معدل لدخول البيانات لكى يعمل هذا النظام فى الزمن المباشر real time .

الفصل الخامس

برمجة المعالج Z 80

Programming The Z80 Microprocessor

5-1 مقدمة

يعتبر المعالج Z80 صورة متطورة ومنقحة للمعالج Intel8085 حيث أن جميع أوامر الشريحة 8085 تتوافق مع الشريحة Z80 ولكن الشريحة Z80 تمتاز ببعض المميزات الأخرى الغير موجودة فى الشريحة 8085 والتى سنراها فى أثناء دراستنا لهذا الجزء . سنتبع فى هذا الفصل نفس الطريقة التى اتبعناها فى الفصل السابق حيث سنقسم أوامر الشريحة Z80 إلى مجموعات وسندرس بالتفصيل من كل مجموعة بعض الأوامر الكثيرة الاستخدام على أننا سنعرض فى نهاية الفصل لجداول مختلفة لجميع الأوامر حيث يمكن للقارىء الرجوع إليها . ولقد تعمدنا أن نتبع نفس طريقة الشرح فى الفصل السابق حتى يتمكن من يريد أن يتتبع المعالج Z80 فقط دون اللجوء إلى مراجعة أى معالج آخر قد لا يحتاج إليه فى أثناء تدريبه ، كما أننا سنعرض مقارنات بسيطة بين البروسيسور Z80 و البروسيسور 8085 فى بعض المواقف التى تتطلب ذلك حتى تكتمل الفائدة لمن يريد ذلك .

5-2 مجموعة أوامر الانتقال
Transfer instructions

يقوم أى أمر من أوامر هذه المجموعة بنقل معلومة (بايت) من مكان لآخر حيث المكان الذى تخرج منه المعلومة سنسميه المصدر Source وسنرمز له أحيانا بالرمز sss وهذا المكان قد يكون مسجلا داخل شريحة المعالج وقد يكون بايتا من بايتات الذاكرة ، وأما المكان الذى ستذهب إليه المعلومة فسوف نسميه الهدف Destination وسنرمز له أحيانا بالرمز ddd وهذا المكان أيضا قد يكون مسجلا داخل شريحة المعالج وقد يكون بايتا من بايتات الذاكرة كما سنرى . تمتاز جميع أوامر الانتقال الخاصة بالشريحة Z80 بأن لها نفس الأحرف مهما كان مصدر أو هدف المعلومة وهذه الأحرف هى LD التى هى اختصارا لكلمة LOAD أو حمل ، وذلك على العكس من الشريحة 8085 التى تستخدم عددا أكثر من الأحرف والتى منها MOV و MVI و LDA وغير ذلك من الصور التى تستخدم كل منها فى حالته الخاصة كما رأينا فى الفصل السابق ، لذلك سنتناول أوامر الانتقال فى حالة الشريحة Z80 على حسب مصدر وهدف المعلومة كما يلى :

5-2-1 نقل معلومة من مسجل إلى مسجل آخر

الصورة العامة لهذا الأمر هى :

LD ddd,sss

ddd مسجل ⟵ sss مسجل

ومعناه تحميل المسجل ddd بمحتويات المسجل sss ، لاحظ أن الذى يتم نقله من المسجل sss هو صورة من المحتويات فقط أما محتويات المسجل فتظل كما هى لا تتغير . من الأمثلة على ذلك ما يلى :

- الأمر LD A,B حيث سيقوم هذا الأمر بنقل (نسخ) محتويات المسجل B (المصدر) إلى المسجل A (الهدف) .
- الأمر LD C,H سيقوم بتحميل المسجل C بمحتويات المسجل H .
- الأمر LD B,B سيقوم بتحميل المسجل B بمحتويات المسجل B . لاحظ أن تأثير هذا الأمر يكافىء تماما "لا تعمل شيئا" وهذه العملية تكون مهمة جدا فى الكثير من التطبيقات ولذلك فقد أفرد لها أمر خاص بها وهو الأمر NOP أى No Operation أو لا تعمل شيئا ، وهذا الأمر سنراه فى الكثير من التطبيقات القادمة إن شاء الله . الشفرات الستعشرية لجميع أوامر الانتقال بين جميع المسجلات بعضها البعض يوضحها شكل (5-1) . بالنظر لهذا الشكل نجد أنه إذا أردنا مثلا نقل محتويات المسجل A إلى المسجل L نستخدم الأمر LD L,A الذى شفرته من شكل (5-1) هى 6F . جميع أوامر نقل المعلومة من مسجل إلى آخر تتكون من بايت واحدة فقط تسمى OP Code وهى اختصار Operation Code أو شفرة العملية .

5-2-2 تحميل مسجل بمعلومة فورية أو ثابت Constant

فى الكثير من التطبيقات نحتاج لتحميل مسجل من المسجلات بثابت معين ، فى هذه الحالة تكون الصورة العامة لمثل هذه الأوامر كما يلى :

LD ddd,data8

ddd المسجل ⟵ data8

ومعناه حمل المسجل ddd بالمعلومة الفورية أو الثابت data8 ، لاحظ أن data8 يقصد بها ثابت مكون من ثمانية بتات ، جميع أوامر هذه المجموعة تتكون من اثنين بايت واحدة هى شفرة الأمر Op Code والثانية هى البايت data8 أو الثابت. شكل (5-2) يبين الشفرات الستعشرية لعملية تحميل أى مسجل من المسجلات بثابت data8 . من هذا الشكل نلاحظ أنه لتحميل المسجل D بالقيمة 55H مثلا نستخدم الأمر LD D,55H والذى ستكون شفرته الستعشرية كما يلى :

16

55H

حيث 16 (البايت الأولى) هى op code كما فى شكل (5-2) أما البايت الثانية فهى قيمة الثابت المراد تحميله فى المسجل D وهو 55H ، لاحظ أن H بعد الرقم تعنى أن هذا الرقم ممثل فى النظام الستعشرى .

مسجل الهدف Destination register	مسجل المصدر Source register						
	A	B	C	D	E	H	L
A,	7F	78	79	7A	7B	7C	7D
B,	47	40	41	42	43	44	45
C,	4F	48	49	4A	4B	4C	4D
D,	57	50	51	52	53	54	55
E,	5F	58	59	5A	5B	5C	5D
H,	67	60	61	62	63	64	65
L,	6F	68	69	6A	6B	6C	6D

شكل (5-1) الشفرة الستعشرية لجميع أوامر الإنتقال بين المسجلات

A	B	C	D	E	H	L
3E data8	06 data8	0E data8	16 data8	1E data8	26 data8	2E data8

شكل (5-2) الشفرات الستعشرية لأوامر تحميل المسجلات بمعلومة فورية أو ثابت data8

مثال 5. 1

المطلوب تحميل المسجلات H, E, D, C, B, A بالمعلومات الفورية أو الثوابت الآتية : 06, 05, 04, 03, 02, 01 وبعد ذلك يتم إجراء إزاحة دورانية لهذه المحتويات بحيث تذهب محتويات المسجل A إلى المسجل B وتذهب محتويات المسجل B إلى المسجل C وهكذا إلى أن تذهب محتويات المسجل H إلى المسجل A . شكل (4-2) فى الفصل السابق يبين رسما توضيحيا ومخطط السير لهذا البرنامج ، أما البرنامج بلغة الأسمبلى والشفرات الستعشرية فإنه موضح فى شكل (5-3) . شكل (4-3) فى الفصل السابق يبين نفس هذا البرنامج مكتوبا بلغة الأسمبلى الخاصة بالشريحة 8085 وبمقارنة الشفرات الستعشرية لكلا البرنامجين فى الشكلين (5-3 و 4-3) نجد أن هناك تطابقا تاما فى الشفرات الستعشرية فى الحالتين وهذا يبين مدى التطابق بين الشريحتين Z80 و 8085 فإن أى برنامج مكتوبا بالشفرات الستعشرية للشريحة 8085 يمكن تنفيذه باستخدام الشريحة Z80

وأما البرامج المكتوبة بالشفرات الستعشرية للشريحة Z80 فليس بالضرورة أنه يمكن تنفيذه على الشريحة 8085 وذلك لأن الشريحة Z80 تحتوى على عدد أكثر من الأوامر الغير معرفة لدى الشريحة 8085 .

العناوين	شفرات أسمبلى	شفرات ستعشرية
E000	LD A,01	3E
		01
E002	LD B,02	06
		02
E004	LD C,03	0E
		03
E006	LD D,04	16
		04
E008	LD E,05	1E
		05
E00A	LD H,06	26
		06
E00C	LD L,A	6F
E00D	LD A,H	7C
E00E	LD H,E	63
E00F	LD E,D	5A
E010	LD D,C	51
E011	LD C,B	48
E012	LD B,L	45

شكل (5-3) برنامج الأزاحة الدورانية

لتنفيذ البرنامج الموجود فى شكل (5-3) فإنه يمكن أن ندخل فى الذاكرة RAM ونكتب الشفرات الستعشرية ابتداء من العنوان E000 وبعد الانتهاء من كتابة البرنامج فى الذاكرة ننفذه باستخدام الأمر GO E000 . أما إذا كان الميكروكومبيوتر الذى نستخدمه به الأسمبلر الخاص بالمعالج Z80 فإنه يمكننا فى هذه الحالة كتابة البرنامج بلغة الأسمبلى مباشرة ثم تنفيذه وسنترك تفاصيل عملية إدخال البرنامج لأنها تختلف من شخص لآخر ولكننا ننصح أن تكتب البرامج الأولى فى مرحلة التدريب بالشفرات الستعشرية ثم بعد ذلك تكتب بلغة الأسمبلى وننصح أيضا أن يتم تنفيذ البرامج الأولى بطريقة الخطوة خطوة حتى يتمكن

المتدرب من ملاحظة تأثير كل أمر على حده ومتابعة تحميل المسجلات وانتقال البيانات من مسجل لآخر .

يمكن أيضا تحميل زوج من المسجلات بمعلومة مكونة من 16 بتا كما يلى :

LD rp,data16

rp زوج المسجلات ← data16

حيث rp ترمز لأى زوج من أزواج المسجلات المتاحة فى المعالج Z80 وهى IY, IX, SP, HL, DE, BC (راجع شكل (2-6) لترى أزواج المسجلات المتاحة فى الشريحة Z80) . شكل (5-4) يبين الشفرات الستعشرية المصاحبة لكل زوج فى هذا الأمر . لاحظ أن جميع هذه الأوامر ستتكون من ثلاث بايتات ، واحدة (الأولى) ستكون شفرة الأمر op code والاثنتان التاليتان ستحتويان المعلومة data16 المكونة من 16 بت كما ذكرنا فيما عدا فى حالة الزوج IX و IY فإن الأمر سيتكون من أربعة بايتات ، إثنتان لشفرة الأمر op code واثنتان للمعلومة data16 . لاحظ أننا مجازا فقط فى هذا المكان نطلق على المسجلات SP, IY, IX كلمة أزواج ولكن ليكن راسخا فى العلم أن كل واحدة منها عبارة عن مسجل واحد مكون من 16 بت ولا يمكن تقسيمه إلى مسجلين كما هو الحال فى الزوج BC مثلا الذى يمكن استخدامه كمسجلين B و C . كمثال على ذلك فإن الأمر :

LD HL,E100

سيحمل الزوج HL بالمعلومة E100H حيث ستذهب البايت الأولى من المعلومة (00) إلى المسجل L والبايت الثانية من المعلومة (E1) ستذهب إلى المسجل H .

الشفرات الستعشرية لهذا الأمر ستكون كالتالى :

21
00
E1

3-2-5 نقل معلومة من مسجل إلى الذاكرة والعكس

لنقل معلومة من مسجل فى داخل المعالج Z80 إلى أى مكان فى الذاكرة أو العكس يمكن استخدام طريقة من ثلاث طرق متاحة لهذا الغرض وهى كالتالى :

BC	DE	HL	SP	IX	IY
01	11	21	31	DD21	FD21
data16	data16	data16	data16	data16	data16

شكل (5-4) الشفرات الستعشرية لأوامر تحميل أزواج المسجلات بمعلومة فورية أو ثابت data16

5-2-3-1 الطريقة المباشرة Direct addressing

في هذه الطريقة يكون عنوان البايت المراد التعامل معها موجودا فى الأمر نفسه (في البايت الثانية والثالثة) ولذلك فإن مثل هذه الأوامر تتكون دائما من ثلاث بايتات واحدة هى شفرة الأمر op code واثنتان للعنوان المراد التعامل معه. هناك أمران فقط تحت هذه المجموعة ، أمر خاص بنقل المعلومات من مسجل التراكم إلى الذاكرة والآخر خاص بنقل المعلومات من الذاكرة إلى مسجل التراكم A ولذلك فاننا نلاحظ أن التعامل بالطريقة المباشرة لا يكون إلا بين مسجل التراكم فقط والذاكرة ، فإذا أردنا نقل معلومة مثلا من المسجل B إلى الذاكرة بهذه الطريقة فإننا ننقل المعلومة أولا إلى المسجل A ثم منه إلى الذاكرة . الأمر الأول فى هذه المجموعة هو الأمر :

LD A,(addr)

A المسجل ⟵ (addr)

الشفرة الستعشرية لهذا الأمر هى :

3A

البايت ذات القيمة الصغرى من العنوان addr

البايت ذات القيمة العظمى من العنوان addr

حيث سيقوم هذا الأمر بنقل محتويات بايت الذاكرة التى عنوانها فى الاثنين بايت الثانية والثالثة فى الأمر نفسه إلى مسجل التراكم A . ونؤكد هنا على كلمة محتويات حتى لا يظن البعض أن العنوان نفسه هو الذى يوضع فى المسجل A كما يوحى شكل الأمر لأول وهلة ولقد تم وضع قوسين حول كلمة addr فى الصورة الحرفية للأمر للتأكيد على ذلك ولأنه بدون هذين القوسين لن يستطيع الأسمبلر التمييز بين ما إذا كان الرقم addr عنوانا أم معلومة فورية مطلوب تحميلها فى المسجل A . كمثال على ذلك انظر إلى الأمر :

LD A,(E100)

حيث يقوم هذا الأمر بتحميل المسجل A بمحتويات العنوان E100 من الذاكرة . الأمر الثانى من أوامر هذه المجموعة هو الأمر :

LD (addr),A

(addr) ⟵ A المسجل

والشفرة الستعشرية لهذا الأمر هى :

32

البايت ذات القيمة الصغرى من العنوان addr

البايت ذات القيمة العظمى من العنوان addr

حيث سيقوم هذا الأمر بنقل محتويات المسجل A إلى بايت الذاكرة التى عنوانها موجود فى البايت الثانية والثالثة من الأمر نفسه ، أى أن هذا الأمر يقوم بالعملية العكسية للأمر السابق . كمثال على ذلك انظر إلى الأمر :

LD (E100),A

والذى شفرته الستعشرية ستكون :

32
00
E1

حيث سيقوم هذا الأمر بتخزين محتويات المسجل A فى بايت الذاكرة التى عنوانها E100 . بذلك نكون قد انتهينا من الطريقة المباشرة لعنونة أو التعامل مع الذاكرة . إن هذه الطريقة كما رأينا مثلها مثل أن تقول لصديقك أحمد ياأخى ياأحمد وأنت مسافر إلى الرياض أرجوك أن تعطى هذا الخطاب لأخى محمد هناك وعنوانه موجود على الخطاب مباشرة ، هنا صديقك أحمد الذى يمثل المعالج الذى سينفذ الأمر سيعرف عنوان أخيك فى الرياض من على الخطاب مباشرة حيث الخطاب فى هذه الحالة يعتبر الأمر المطلوب تنفيذه .

2-3-2-5 الطريقة غير المباشرة Indirect addressing

في هذه الطريقة لا يكون عنوان البايت المراد التعامل معها في الذاكرة موجودا مباشرة في الأمر نفسه ولكنه يكون موجودا في مكان آخر وعلى المعالج الذهاب إلى هذا المكان أولا وقبل تنفيذ الأمر لمعرفة العنوان . هذا المكان يكون زوجا من المسجلات ولابد أن يكون زوجا لأن العنوان كما نعرف دائما يتكون من 16 بت ، وعادة يكون هذا الزوج هو الزوج HL . الصورة العامة للأمر فى هذه الحالة بلغة الأسمبلى تكون كما يلى :

LD ddd,(HL)

المسجل ddd ← (HL)

حيث سيقوم هذا الأمر بنقل محتويات بايت الذاكرة التى عنوانها فى زوج المسجلات HL إلى المسجل ddd . الصورة العكسية لهذا الأمر هى :

LD (HL),sss

(HL) ← المسجل sss

حيث سيقوم هذا الأمر بنقل محتويات مسجل المصدر إلى بايت الذاكرة التى يوجد عنوانها فى زوج المسجلات HL . شكل (5-5) يبين الشفرات الستعشرية لهذه الأوامر فى حالة استعماله مع جميع المسجلات الموجودة فى الشريحة Z80 وكما ذكرنا فإن العنوان لابد وأن يكون موجودا فى الزوج HL . هذه الأوامر (أوامر الطريقة غير المباشرة) ستكون جميعها مكونة من بايت واحدة فقط وهى شفرة الأمر op code . كمثال على ذلك الأمر LD B,(HL) والذى ستكون شفرته الستعشرية 46 حيث سيقوم هذا الأمر بنقل محتويات البايت التى عنوانها فى الزوج HL إلى المسجل B . لاحظ وجود القوسين حول الزوج HL فى جميع

هذه الأوامر للدلالة على أن المقصود هو محتويات العنوان الموجـــود فـــى HL وليس القيمة الموجودة فى HL مباشرة .

	A	B	C	D	E	H	L
ddd ← (HL)	7E	46	4E	56	5E	66	6E
(HL) ← sss	77	70	71	72	73	74	75

شكل (5-5) الشفرات الستعشرية لأوامر الانتقال بين المسجلات والذاكرة بالطريقة غير المباشرة

هناك ميزة يمتاز بها المسجل A عن باقى مسجلات الشريحة فى حالة التعـــامل بينه وبين الذاكرة بالطريقة غير المباشرة وهى أن عنوان البايت المراد التعـــامل معها فى هذه الحالة يمكن وضعه فى أى زوج من أزواج المســجلات الأخـــرى وليس بالضرورة الزوج HL كما سبق ، وذلك كما قلنا كحالة خاصة فقط للمسجل A . شكل (5-6) يبين الشفرة الستعشرية والأسمبلى وما يقوم به كل واحد مـــن هذه الأوامر .

الشفرة الأسمبلى	الشفرة الستعشرية	ما يفعله الأمر
LD A,(BC)	0A	A ← (BC)
LD A,(DE)	1A	A ← (DE)
LD (BC),A	02	(BC) ← A
LD (DE),A	12	(DE) ← A

شكل (5-6) أوامر الانتقال بين المسجل A والذاكرة بالطريقة غير المباشرة

كما رأينا فإن الطريقة غير المباشرة فى التعامل مع الذاكرة مثلها مثل أن تقـــول لصديقك أحمد ياأخى ياأحمد وأنت مسافر إلى الرياض خذ هذا الخطاب وأعطــه لأخى محمد ولكن أرجوك قبل سفرك أن تمر على والدى لتعرف منــــه عنـــوان أخى محمد فى الرياض لأنى لا أعرفه . هنا صديقك أحمد يمثل المعـــالج الـــذى سيقوم بالتنفيذ والوالد يمثل المسجلين HL حيث يحتويان العنوان المراد التعـــامل معه واللذان لابد من المرور عليهما قبل تنفيذ الأمر . السؤال الآن كيف نضـــع عنواناً ما فى زوج من المسجلات ؟ إن هذه العملية بسيطة جدا حيـــث يســتخدم فيها أوامر تحميل زوج مسجلات بمعلومة فورية أو ثابت مكون مــــن 16 بـــت والتى درسناها فى الجزء السابق (5-2-2 تحميل مسجل بمعلومة فورية) .

5-2-3-3 طريقة الفهرسة لعنونة الذاكرة Indexed addressing

هناك مسجلان لم نتكلم عنهما تفصيليا إلى الآن وهما المسجلان IX و IY وكل منهما يتكون من 16 بت . هذان المسجلان يختلفان عن أزواج المسجلات الأخرى فى أنه لا يمكن استخدام أى منهما كمسجلين 8 بتات منفصلين ولكن كل منهما يستخدم كمسجل 16 بت مثله فى ذلك مثل مسجل عداد البرنامج program counter أو مؤشر المكدسة stack pointer . هذان المسجلان يستخدمان فى عملية الإشارة إلى بايتات الذاكرة تماما مثل الزوج HL ولكن بإمكانيات أكثر ، ولكى نفهم ذلك انظر إلى الأمر التالى :

LD B,(IX+5)

هذا الأمر سيقوم بتحميل المسجل B بمحتويات بايت الذاكرة التى عنوانها عبارة عن حاصل جمع محتويات المسجل IX زائد خمسة . لكى نوضح هذا الأمر افترض الوضع التالى :

B	IX	E105
05	E100	66

بعد تنفيذ الأمر LD B,(IX+5) سيصبح الوضع كالتالى :

B	IX	E105
66	E100	66

الرقم الذى يضاف على محتويات المسجل IX أو المسجل IY يسمى "الإزاحة" displacement وهذه الإزاحة تشغل بايت كاملة ولذلك فإن قيمتها ستتراوح بين +127 و -128 حيث الإزاحة الموجبة معناها إضافة إلى محتويات مسجل الفهرسة (IX, IY) وأما الإزاحة السالبة فمعناها طرح من مسجل الفهرسة . الشفرات الستعشرية لجميع هذه الأوامر موجودة فى جداول الأوامر الموضحة فى نهاية هذا الفصل .

5-3 تمارين

استخدم قائمة أوامر الانتقال الخاصة بالشريحة Z80 لحل التمارين الموجودة فى الجزء 4-3 فى الفصل السابق .

5-4 مجموعة أوامر الحساب
Arithmatic Instructions

سندرس فى هذا الجزء بعض الأوامر التى تقوم بإجراء العمليات الحسابية الأولية وهى الجمع والطرح ، وكما علمنا من قبل فإن مسجل التراكم لابد وأن يكون طرفا فى أى عملية من هذه العمليات كما أن نتيجة هذه العملية سواء كانت جمعا أو طرحا تكون دائما موجودة فى مسجل التراكم A . هناك أيضا خاصية مهمة فى مجموعة أوامر الحساب (ومثلها أيضا أوامر المنطق كما سنرى) وهى أنه نتيجة تنفيذ أى أمر من هذه الأوامر فإن الأعلام الموجودة في مسجل الحالة Status Register تتأثر بهذه النتيجة . راجع مسجل الحالة ومحتوياته ومتى يكون أى علم من أعلامه يساوى صفرا أو واحدا وذلك فى الفصل الثانى . كما ذكرنا فإن طرفا من طرفى العملية الحسابية أو معامل من معامليها لابد وأن يكون موضوعا فى المسجل A وأما الطرف الثانى أو المعامل الثانى فإن مصدره سيكون واحدا من أربعة أماكن موضحة كالتالى :

الرمز المستخدم	مصدر المعامل الثانى للعملية الحسابية
reg	مسجل 8 بت من مسجلات المعالج .
(HL)	بايت من بايتات الذاكرة عنوانها فى HL .
data8	ثابت أو معلومة فورية مكونة من 8 بت .
(IX+d) (IY+d)	بايت من بايتات الذاكرة معنونة بطريقة الفهرسة باستخدام المسجلين IX أو IY

5-4-1 الأمران ADD و SUB

الصورة العامة لأمر الجمع ADD هى :

ADD A,reg

A ⟵ A + reg

حيث سيقوم هذا الأمر بجمع محتويات المسجل reg مع محتويات المسجل A ووضع النتيجة فى المسجل A مع التأثير على الأعلام لاحظ أننا فى حالة

أسمبلر الشريحة 8085 كنا نكتب هذا الأمر ADD reg بدون ذكر المسجل A على أساس أنه بديهى أن المعامل الثانى للعملية يكون فى المسجل A ، أما فى أسمبلر الشريحة Z80 فلابد من ذكر طرفى العملية الحسابية بالرغم من أن أحدهما يكون دائما المسجل A وإن كانت بعض المراجع تهمل ذلك . الصور العامة الأخرى لأمر الجمع ستكون كالتالى :

ADD A,(HL)

حيث سيقوم هذا الأمر بجمع محتويات بايت الذاكرة التى يوجد عنوانها فى الزوج HL مع محتويات المسجل A وتوضع النتيجة فى المسجل A .

ADD A,data8

حيث سيجمع الثابت data8 مع المسجل A وتوضع النتيجة فى المسجل A .

ADD A,(IX+d)
ADD A,(IY+d)

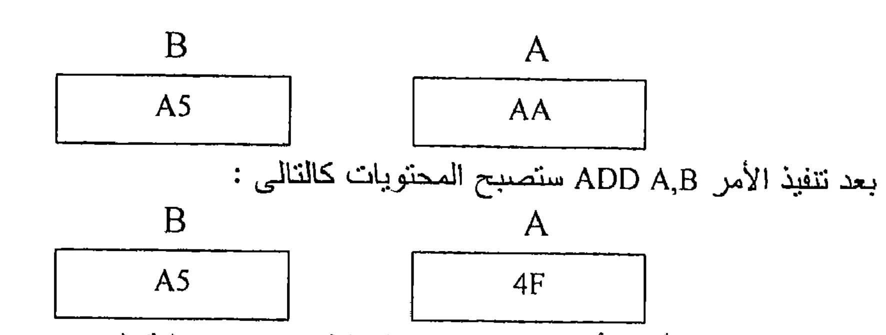
حيث سيجمع محتويات المسجل A مع محتويات بايت الذاكرة التى عنوانها عبارة عن محتويات المسجل IX أو IY زائد الإزاحة d وتوضع النتيجة فى المسجل A . بنفس الطريقة يمكن كتابة الصورة العامة لأمر الطرح تبعا لمصدر المعامل الثانى كما يلى :

SUB A,reg
A⟵ A - reg
SUB A,(HL)
A⟵ A - (HL)
SUB A,data8
A⟵ A - data8
SUB A,(IX+d)
SUB A,(IY+d)
A⟵ A - (IX+d)
A⟵A- (IY+d)

فى جميع هذه الأوامر يتم طرح المعامل الثانى (data8, (HL), reg, (IY+d),(IX+d)) من المسجل A وتوضع النتيجة فى المسجل A ، أى أننا نؤكد هنا على أن المطروح منه يكون دائما المسجل A . لاحظ أن المسجل A يذكر مع هذا الأمر أيضا وذلك بالطبع لا دخل للمستخدم فيه ولكنه من الشروط التى يفرضها الأسمبلر (كما فى بعض المراجع) ، سنهمل ذكر الشفرات الستعشرية للأوامر ابتداء من هذا الموضع ومن يريد التعرف عليها أو استخدامها فعليه الاستعانة بالأشكال الخاصة بالأوامر فى نهاية هذا الفصل .

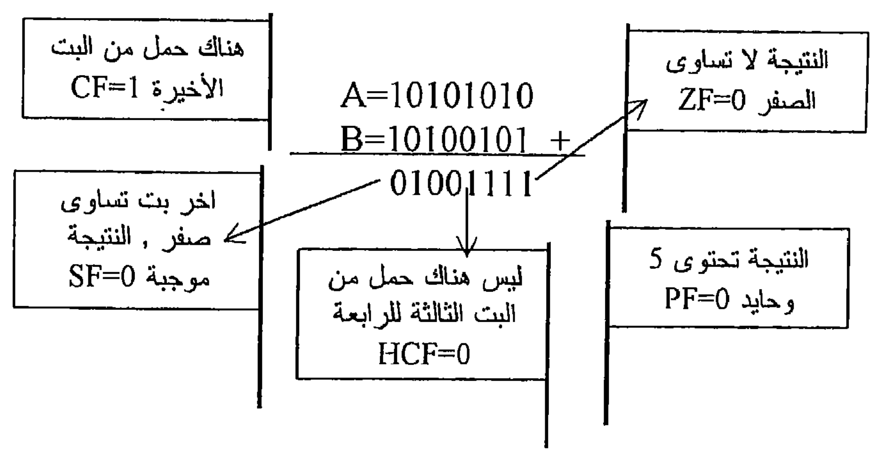
مثال 5-2

افترض المحتويات الآتية للمسجلين A و B قبل تنفيذ الأمرين ADD و SUB :

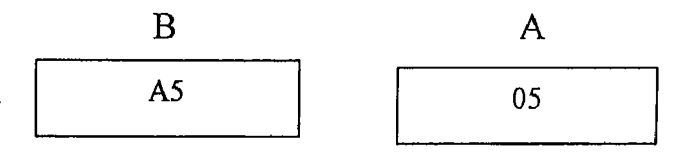


بعد تنفيذ الأمر ADD A,B ستصبح المحتويات كالتالى :

ولكى نرى كيفية تأثر الأعلام بنتيجة هذه العملية سنجرى عملية الجمع على الشفرات الثنائية لكل من الرقمين كما يلى :



الآن افترض أننا نفذنا أمر الطرح SUB B على المحتويات الأولى للمسجلين أى A=AAH و B=A5H فإنه بعد تنفيذ هذا الأمر ستصبح محتويات المسجلين كالتالى:



ولكى نرى كيف تمت عملية الطرح وكيف تأثرت الأعلام سنجرى عملية الطرح على الشفرات الثنائية لمحتويات المسجلين A و B كالتالى :
كما نعلم فإن عملية الطرح الثنائى يتم تحويلها إلى عملية جمع حيث سنجمع محتويات المسجل A (المطروح منه) مع المتمم الثنائى لمحتويات المسجل B (المطروح) (انظر الملحق الأول فى نهاية الكتاب لمراجعة عمليات الجمع والطرح الثنائى) . المتمم الثنائى لمحتويات المسجل B (10100101) هو 01011011 وبذلك تصبح عملية الطرح عملية جمع كالتالى :

A = 10101010

المتمم الثنائى لمحتويات المسجل B = 01011011 +

00000101

وبناء على ذلك ستكون الأعلام كالتالى :

- طالما أن البت الأولى لا تساوى صفرا فالنتيجة لا تساوى صفرا ويكون علم الصفر ZF=0 .
- تحتوى النتيجة على عدد زوجى من الوحايد (اثنين) لذلك سيكون علم الباريتى واحدا ، PF=1 .
- هناك حمل من البت الثالثة إلى البت الرابعة فى حالة الجمع لذلك فعلم الحمل النصفى HCF=0 .
- آخر بت (رقم 7) تساوى صفرا ، لذلك فالنتيجة موجبة وعلم الإشارة يكون دائما مساويا لمحتويات آخر بت , إذن SF=0 .
- المفروض فى عمليات الطرح يهمنا أن نعرف إذا كان هناك استلاف أم لا لأنه فى عملية الطرح لن يكون هناك حمل بما أن عملية الطرح قد حولت إلى عملية جمع لذلك فإنه إذا كان هناك حمل فى عملية الجمع فان ذلك يعنى أنه لن يكون هناك استلاف فى عملية الطرح وسيكون علم الحمل CYF=0 وهى الحالة التى معنا الآن والعكس صحيح إذا لم يكن هناك حمل فى عملية الجمع . وهذا هو ما طبقناه فى حالة العلم HCF

2-4-5 الأمران ADC و SBC

بالنسبة للأمر ADC فإنه يجمع المعامل الثانى سواء كان فى مسجل أو ذاكرة أو قيمة فورية مع محتويات المسجل A مع محتويات علم الحمل CY (صفر أو واحد) ويضع النتيجة فى المسجل A . الصورة العامة لهذه الأوامر وعلى حسب مصدر المعامل الثانى ستكون كما يلى :

```
ADC A,reg
A <—— A + CY + reg
ADC A,(HL)
A <—— A + CY + (HL)
ADC A,data8
A <—— A + CY + data8
ADC A,(IX+d)
A <—— A + CY + (IX+d)
ADC A,(IY+d)
A <—— A + CY + (IY+d)
```

يمكننا الآن تكرار نفس القول بالنسبة لأمر الطرح SBC حيث يقوم هذا الأمر بطرح المعامل الثانى سواء كان فى مسجل أو ذاكرة أو قيمة فورية مع محتويات

علم الحمل CY (صفر أو واحد) من المسجل A ثم توضع نتيجة الطرح في المسجل A ، نؤكد هنا على أن المطروح منه دائما هو المسجل A . الصورة العامة لهذه الأوامر وعلى حسب مصدر المعامل الثانى ستكون كما يلى :

SBC A,reg

A⟵A - CY - reg

SBC A,(HL)

A⟵A - CY - (HL)

SBC A,data8

A⟵ A - CY - data8

SBC A,(IX+d)

A⟵A - CY - (IX+d)

SBC A,(IY+d)

A⟵ A - CY - (IY+d)

مثال 5-3

```
E000 E001        LD C,F9H
E002 E003        LD B,23H
E004 E005        LD E,35H
E006 E007        LD D,9AH
E008             LD A,C
E009             ADD A,E
E00A E00B E00C   LD (E100),A
E00D             LD A,B
E00E             ADC A,D
E00F E010 E011   LD (E101),A
E012 E013        LD A,00
E014             ADC A,A
E015 E016 E017   LD (E102),A
```

شكل (5-7) برنامج المثال 5-3

المطلوب جمع الرقمين 23F9H و 9A35H ووضع نتيجة الجمع في أماكن الذاكرة E100 و E101 و E102 .

هذا المثال هو المثال رقم 4-11 وقد سبق حله كتطبيق على أمر الجمع مع الحمل فى حالة الشريحة 8085 ويبين شكل (4-6) رسما توضيحيا ومخطط السير والبرنامج لطريقة حل هذا المثال ، لذلك يمكن مراجعته أولا وسنعيد كتابة البرنامج فقط بلغة الأسمبلى الخاصة بالشريحة Z80 فى شكل (5-7) . قبل أن نترك أوامر الجمع والطرح يجب أن نفهم جيدا متى يكون من الضرورى

استخدام الأمر ADC ؟ ومتى يكون من الضرورى عدم استخدامه؟ مثـــلا فـــى المثال السابق كان من الضرورى عدم استخدام الأمر ADC فى عمليـــة الجمـــع الأولى (E + C) ولكن فى هذه الحاله لابد من استخدام الأمر ADD خوفا مـن أن يكون علم الحمل CY به واحد من أى عملية سابقة ونحن لا ندرى فيجمـــع مـــع عملية الجمع الأولى إذا استخدمنا الأمر ADC وتكون النتيجة خاطئة . أما فـــى عملية الجمع الثانية (D + B + CY) فإنه لابد من استخدام الأمر ADC لأننا نريد هنا أن نأخذ قيمة علم الحمل فى الاعتبار .

5-4-3 الأمران INC و DEC

هذان الأمران يستخدمان لزيادة أو إنقاص واحد على أو من محتويات مســجل أو بايت من بايتات الذاكرة . الصورة العامة للأمـــر INC وعلـــى حســب مكـــان المعلومة يمكن كتابتها كالتالى :

INC reg
reg ⟵ reg + 1
INC (HL)
(HL) ⟵ (HL) + 1
INC (IX+d)
(IX+d) ⟵ (IX+d) + 1
INC (IY+d)
(IY+D) ⟵ (IY+d) + 1

بنفس الطريقة يمكن كتابة الصورة العامة للأمر DEC كما يلى :

DEC reg
reg ⟵ reg - 1
DEC (HL)
(HL) ⟵ (HL) – 1
DEC (IX+d)
(IX+D) ⟵ (IX+d) - 1
DEC (IY+d)
(IY+d) ⟵ (IY+d) - 1

5-4-4 العمليات الحسابية على أزواج المسجلات

عند إجراء العمليات الحسابية على أزواج المســجلات يلعـــب الـــزوج HL دور مسجل التراكم من حيث أن المعامل الأول فى العملية الحسابية لابد وأن يكـــون فى الزوج HL ونتيجة العملية الحسابية تذهب دائما إلى الزوج HL . يجـــب أن نعلم أنه عند إجراء العمليات الحسابية على أزواج المسجلات أن الأعلام لا تتأثر

بهذه العمليات فى الكثير من الأحيان ويجب النظر فى حالة كل أمر منفصلـة .
الصورة العامة للأمرين INC و DEC فى هذه الحالة هى :

INC rp
rp⟵ rp + 1
DEC rp
rp⟵rp −1

حيث rp ترمز لأى زوج من أزواج المسجلات SP, HL, DE, BC أو مسجل من المسجلات ال 16 بت وهى المسجل IX أو المسجل IY . كأمثلة على ذلك انظـر إلى الأوامر التالية :

INC HL
DEC SP
INC IX

الأمر الأول سيزيد واحدا على محتويات المسجلين HL والثانى سينقص واحـــدا من محتويات المسجل SP والثالث سيزيد واحدا على محتويـــات المســجل IX .
الأوامر INC rp و DEC rp ليس لها تأثير على الأعلام .
من أوامر الجمع والطرح التى تجرى على أزواج المسجلات ما يلى :

ADD HL,rp
HL⟵HL + rp
ADC HL,rp
HL⟵HL + rp + CY
SBC HL,rp
HL⟵ HL - rp - CY

فى جميع هذه الأوامر ترمز rp لأى زوج من الأزواج SP, HL, DE, BC ما عدا المسجلين IX, IY فلا يمكن استخدامهما مع هذه الأوامر . لاحظ أيضا أن أمـــر الطرح الوحيد المتاح هو أمر الطرح مع الحمل SBC ولذلك فإنه عند إجــراء أى عملية طرح بدون أخذ علم الحمل فى الحسبان يجب فى هذه الحالة التأكد مـن أن علم الحمل يساوى صفرا . الأمر ADD HL,rp ليس له تأثير على الأعلام وأمـــا الأمران ADC HL,rp و SBC HL,rp فيؤثران على الأعلام ما عدا علم الحمـــل النصفى HC . نؤكد هنا على أن المطروح منه فى الأوامر السابقه هو المســجلين HL . هناك بعض الأوامر التى تسمح للمسجل IX أو المسجل IY بأن يلعب دور مسجل التراكم فى عمليات الجمع على أزواج المسجلات ، وهذه الأوامر هى :

IX⟵ IX + BC	ADD IX,BC
IX⟵IX + DE	ADD IX,DE
IX⟵IX + SP	ADD IX,SP
IX⟵IX + IX	ADD IX,IX
IY⟵IY + BC	ADD IY,BC
IY⟵ IY + DE	ADD IY,DE

IY ⟵ IY + SP   ADD IY,SP
IY ⟵ IY + IY   ADD IY,IY

جميع هذه الأوامر تؤثر على الأعلام ما عدا علم الحمل النصفـــى HC وأيضـــا جميع هذه الأوامر ليس لها نظير لعملية الطرح .

5-4-5 أمر المقارنة Compare Instruction

هناك أمر واحد فقط للمقارنة حيث أن عملية المقارنة لا تجرى على أى معلومـــة مكونة من 16 بت . لكى تتم عملية المقارنة فإن أحد المعاملين لابد وأن يكون فى المسجل A والمعامل الآخر يكون إما فى مسجل من مسجلات المعـــالج أو فـــى بايت من بايتات الذاكرة . عند تنفيذ أمر المقارنة يقوم المعالج بطرح محتويـــات المعامل الثانى من محتويات المسجل A وتهمل نتيجة الطرح تمامـــا ولا تتغـــير محتويات المسجل A نتيجة هذه العملية ولكن الذى يتأثر فقط بهذه العملية هـــو الأعلام . الصورة العامة لأمر المقارنة وعلى حسب مصدر المعامل الثانى يمكن كتابتها كما يلى :

A - reg	CP reg
A - (HL)	CP (HL)
A - data8	CP data8
A - (IX+d)	CP (IX+d)
A - (IY+d)	CP (IY+d)

لاحظ أننا لم نكتب السهم الذى يوضح أين تذهب نتيجة عمليـــة الطـــرح علـــى أساس أن النتيجة تهمل كما ذكرنا . كأمثلة على ذلك انظر إلى الأوامر التالية :

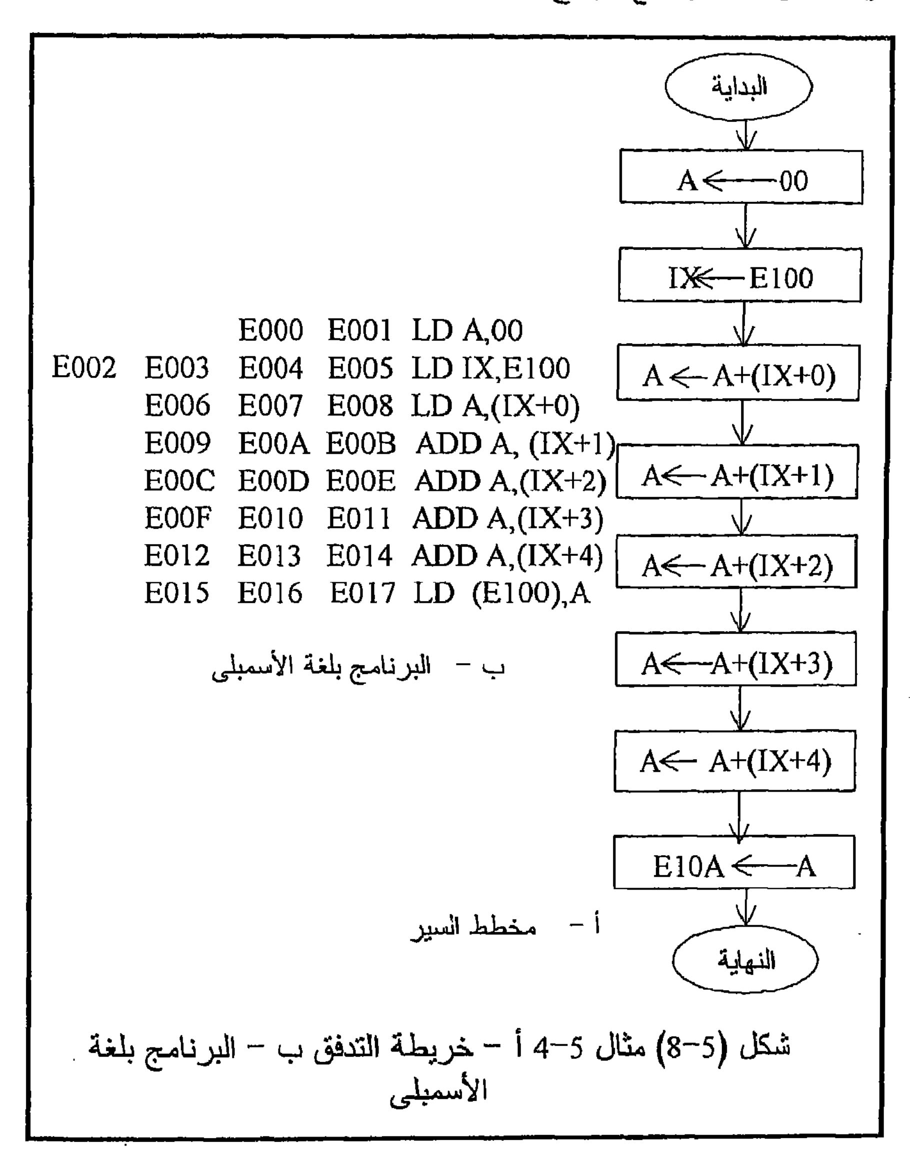
```
CP B
CP 3FH
CP (IY+9)
```

حيث سيقارن الأمر الأول محتويات المســجل B مـــع محتويـــات المســـجل A وسيقارن الأمر الثانى محتويات المسجل A مع الثابت أو القيمة الفوريـــة 3FH وأما الأمر الثالث فسيقارن محتويات المسجل A مع محتويات بايت الذاكرة التـــى عنوانها تسعة زائد محتويات المسجل IY .

مثال 5-4

اكتب برنامجا يجمع محتويات الخمس بايتات E100, E101, E102, E103, E104 ويضع النتيجة فى البايت E10A على اعتبار أن النتيجة المتوقعة لن تزيـــد عـــن بايت واحدة ، أى لن يكون هناك حمل على الإطلاق . هذا المثال من الممكـن أن يكون تدريبا جيدا على طرق الاتصال بالذاكرة التى تمت دراستها حتـــى الآن . شكل (5-8) يبين مخطط السير والبرنامج لهذا المثال مستخدمين طريقة الفهرسـة

مع المسجل IX للاتصال بالذاكرة ، حاول كتابة البرنامج مرة أخرى مســـتخدما الطريقة غير المباشرة مع الزوج HL .



شكل (5-8) مثال 5-4 أ - خريطة التدفق ب - البرنامج بلغة الأسمبلى

5-5 تـمـــاريـــن

حل التمارين الموجودة فى الجزء 5-4 فى الفصل الرابع مستخدما أوامـــر لغـــة الأسمبلى الخاصة بالشريحة Z80 .

5-6 مجموعة أوامر القفز
Jump Instructions

القاعدة العامة أن المعالج يقوم بتنفيذ البرنامج حسب ترتيب الأوامر الموجودة فيه من أول البرنامج إلى نهايته . ولقد كنا حريصين فى جميع الأمثلة السابقة علـــى الحفاظ على هذه القاعدة ، ولكن هناك بعض المواقف أو قل بعـــض التطبيقـــات التى تتطلب الخروج على هذه القاعدة كأن يطلب منك مثلا تنفيذ عمليـــة معينـــة (عدد من الأوامر) عدد معين أو حتى عدد لا نهائى من المرات . فعندما يكـــون المعالج مثلا مراقبا لدرجة الحرارة فى عملية صناعية معينة فإن عليه أن يقـــرأ درجة الحرارة ويقارنها بدرجة حرارة مخزنة فى الذاكـــرة كمرجـــع وإذا زادت الحرارة عن حد معين يقوم المعالج بضرب جرس إنذار ، وإذا نقصت عن حـــد معين يشغل سخان لزيادتها ، مثل هذا البرنامج سيكون عبارة عن مجموعة مـــن الأوامر التى تنفذ إلى مالانهاية طالما أن المعالج يراقب درجة الحرارة . لقد أتاح المعالج هذه العملية بتوفير بعض الأوامر التى تمكنك كمبرمج من القفز بعمليـــة التنفيذ من مكان لآخر خلال البرنامج ، وهناك ثلاثة أنواع مـــن القفـــز توفرهـــا الشريحة Z80 وهى كما يلى :

5-6-1 القفز غير المشروط Unconditional jump

عند تنفيذ أى عملية قفز غير مشروط ينتقل المعالج بعملية التنفيذ إلـــى المكـــان الجديد دون أى قيد أو شرط ، وهناك أمر واحد فقط من أوامر الشـــريحة Z80 يقوم بهذه العملية والصورة العامة لهذا الأمر هى :

JP addr

عند تنفيذ هذا الأمر يوضع العنوان addr الذى سيتم القفز إليه فى عداد البرنـــامج PC فيصبح الأمر الموجود عند هذا العنوان addr هو الأمر الذى عليه الدور فـــى التنفيذ . لاحظ أن هذا الأمر يتكون من ثلاث بايتات واحدة هـــى شـــفرة الأمـــر واثنتان للعنوان addr الذى سيتم القفز إليه . إن القفز باستخدام الأمر JP addr قد يكون إلى الأمام فى البرنامج وقد يكون إلى الخلف . إذا كان القفز إلـــى الأمـــام سينتج عن ذلك وجود جزء من البرنامج لن ينفذ على الإطلاق وهو الجزء الـــذى يقع بين أمر القفز JP addr والأمر الذى سيتم القفز إليه . أما إذا كان القفز إلـــى الخلف فإنه سينتج عن ذلك ما يسمى بالحلقـــة اللانهائيـــة Infinite loop والتـــى سيستمر المعالج فى تنفيذها إلى مالانهاية . شكل (4-8) يبين خريطة تدفق لعملية القفز غير المشروط بنوعيها الأمامى والخلفى .

5-6-2 القفز المشروط Conditional jump

كما يوحى الاسم فإنه فى هذا النوع من القفز لن يتم القفز إلا إذا تحقق شرط معين ، أما إذا لم يتحقق هذا الشرط فإن البرنامج يتم تنفيذه فى التتابع الطبيعى حيث سينفذ الأمر الذى بعد أمر القفز مباشرة . إن شروط القفز توضع دائما على الأعلام التى فى مسجل الحالة SR ، فيمكنك مثلا أن تجعل القفز مشروطا بأن تكون النتيجة صفرا أو تجعله مشروطا بأن تكون النتيجة سالبة وهكذا . حيث أن هناك خمسة أعلام واحد منها وهو علم الحمل النصفى HC لا يستخدم كشرط فى عمليات القفز فإنه يتبقى أربعة أعلام يمكن أن تستخدم فى أوامر القفز المشروط كما يلى :

اقفز إذا كانت النتيجة صفرا	JP Z,addr	ZF=1
اقفز إذا كانت النتيجة ليست صفرا	JP NZ,addr	ZF=0
اقفز إذا كانت النتيجة سالبة	JP M,addr	SF=1
اقفز إذا كانت النتيجة موجبة	JP P,addr	SF=0
اقفز إذا كان هناك حمل	JP C,addr	CF=1
اقفز إذا لم يكن هناك حمل	JP NC,addr	CF=0
اقفز إذا كانت الباريتى زوجية	JP PE,addr	PF=1
اقفز إذا كانت الباريتى فردية	JP PO,addr	PF=0

لاحظ أن عدد هذه الأوامر ثمانية ، إثنان منها لكل علم من الأعلام الأربعة تمثل جميع الحالات التى يمكن أن يكون فيها هذا العلم صفرا أو واحدا . أيضا جميع هذه الأوامر لابد وأن تتكون من ثلاث بايتات واحدة هى شفرة الأمر op code واثنتان للعنوان الذى سيتم القفز إليه . إن النتيجة التى سيتوقف عليها أمر القفز هى آخر نتيجة تأثرت بها الأعلام ، ولذلك فإنه قبل أن نكتب أى أمر من أوامر القفز المشروط يجب أن ندرس جيدا هل الأمر السابق لأمر القفز يؤثر على الأعلام أم لا .

5-6-3 القفز النسبى Relative jump

هناك أنواع أخرى من القفز متاحة لدى المعالج Z80 مثل القفز النسبى والقفز للبرامج الفرعية والعودة منها وسنترك الكلام عن هذه الأنواع حيث سيتم شرحها بالتفصيل فى فصول خاصة بذلك .

مثال 5-5

اكتب برنامجا يقرأ محتويات البايت E100 باستمرار إلى مالانهاية ثم يختبر هذه المحتويات بحيث إذا كانت صفرا يضع واحدا فى المسجل B وإذا كانت سالبة

يضع اثنين فى المسجل B وإذا كانت موجبة يضع أربعة فى نفس المسجل . شكل (4-9) فى الفصل السابق يبين مخطط السير لهذا البرنامج وسنعيد فقط كتابة البرنامج بلغة الأسمبلى الخاصة بالشريحة Z80 كما فى شكل (5-9) .

```
E000 E001 E002    LD HL,E100
     E003 E004    LD A,00
     E005 ADD     A,(HL)
E006 E007 E008    JP NZ,E00E
     E009 E00A    LD B,01
E00B E00C E00D    JP E000
E00E E00F E010    JP P,E016
     E011 E012    LD B,02
E013 E014 E015    JP E000
     E016 E017    LD B,04
E018 E019 E01A    JP E000
```

شكل (5-9) برنامج المثال 5-5

5-7 مهمة أخرى للأسمبلر

المهمة الوحيدة التى عرفناها للأسمبلر حتى الآن هى مهمة تحويل شفرات الأسمبلى إلى شفرات ثنائية أو لغة ماكينة ، ولكن لحسن الحظ فإن هناك مهام أخرى يستطيع الأسمبلر القيام بها ومن شأن هذه المهام أن تريح المبرمج وتوفر عليه الكثير من المجهود . جزء 4-7 فى الفصل السابق تناول هذه المهام كما تناول أيضا عملية تقسيم أى أمر من أوامر لغة الأسمبلى إلى أجزاء مختلفة وكيف يتعرف الأسمبلر على هذه الأجزاء ، لذلك فإننا لن نكرر هذا الجزء هنا ولكن نحيل القارىء لمراجعته فى الفصل السابق مع الأخذ فى الاعتبار الفوارق البسيطة بين شفرات الأسمبلى الخاصة بالشريحة Z80 والشريحة 8085 .

5-8 أوامر الإدخال والإخراج
Input Output Instructions

إلى الآن رأينا كيف نبرمج شريحة المعالج وكيف نحرك المعلومات داخلها من مسجل إلى مسجل آخر ومن أى مسجل إلى الذاكرة والعكس ، ولكن لم نعرف

حتى الآن كيف نظهر معلومة على شاشة عرض مثلا أيا كان نوع هذه الشاشة ، أو كيف ندخل معلومة إلى المعالج من خلال لوحة مفاتيح على سبيل المثال . إن لوحة المفاتيح وشاشة العرض يعتبران مثالين من العديد من الأمثلة التى تحتاج إلى عمليات الإخراج والإدخال . حينما يستخدم المعالج للتحكم فى أى متغير فى عملية صناعية وليكن مثلا درجة الحرارة فإنه لابد من إدخال درجة الحرارة إلى المعالج بعد تهيئتها ووضعها فى الصورة المناسبة لذلك ، وكذلك إذا أراد المعالج رفع درجة حرارة العملية الصناعية أو ضرب جرس إنذار فإنه يخرج إشارة معينة على بوابة إخراج تؤخذ وتهيأ فى الصورة المناسبة للجهاز الذى ستذهب إليه سواء كان سخانا أو جرسا . إن جميع عمليات الإدخال والإخراج تتم من خلال ما يسمى ببوابات الإدخال والإخراج والتعامل مع هذه البوابات دائما ينقسم إلى قسمين : قسم خاص بالبناء الإلكترونى لهذه البوابات وكيفية توصيلها مع المعالج وهذا القسم سندرسه بالتفصيل فى فصل قادم إن شاء الله ، والقسم الآخر هو كيفية برمجة المعالج للتعامل مع هذه البوابات وهو موضوع دراستنا فى هذا الجزء حيث سندرس الأوامر الخاصة بذلك وسنفترض فى دراستنا لهذا الجزء أن القارىء لديه على الأقل بوابة إدخال input port وبوابة إخراج output port موصلين فى الميكروكومبيوتر الذى يستخدمه فى التدريب وكتابة البرامج .

1-8-5 أوامر الأدخال Input Instructions

الصورة العامة لأمر الإدخال هى :

IN A, no

محتويات البوابة رقم .no ← المسجل A

حيث IN هى اختصار لكلمة Input بمعنى ادخل ، وسيقوم هذا الأمر بإدخال المعلومة الموجودة على بوابة الإدخال رقم .no إلى مسجل التراكم A . لاحظ أن عملية الإدخال بهذا الأمر تكون دائما على المسجل A حيث يمكن نقل المعلومة بعد ذلك إلى أى مكان آخر . هذا الأمر يتكون من اثنين بايت ، واحدة هى شفرة الأمر op code والأخرى هى رقم البوابة التى سيتم التعامل معها . ولذلك فإنه طالما أن رقم البوابة يشغل بايت كاملة فإن ذلك يعنى أنه يمكن التعامل مع 2^8 =256 بوابة إدخال تبدأ من البوابة رقم 00H إلى البوابة رقم FFH . انظر شفرة هذا الأمر فى جداول الأوامر فى نهاية هذا الفصل . هناك طريقة غير مباشرة للتعامل مع بوابات الإدخال والصورة العامة لها كالتالى :

IN reg,(C)

بوابة الأدخال التى رقمها فى المسجل C ← المسجل reg

حيث سيقوم هذا الأمر بإدخال المعلومة الموجودة فى بوابة الإدخال التى رقمها فى المسجل C إلى المسجل reg الذى هو أى مسجل من مسجلات المعالج وهذه ميزة عظيمة لم تكن موجودة فى المعالج 8085 .

5-8-2 أوامر الأخراج Output Instruction

الصورة العامة لأمر الأخراج هى :

OUT no ,A

no. البوابة رقم ← المسجل A

حيث OUT هى اختصار لكلمة Output التى تعنى إخرج ، وسيقوم هذا الأمر بإخراج المعلومة الموجودة فى المسجل A إلى بوابة الإخراج التى رقمها .no . هذا الأمر أيضا يشغل اثنين بايت واحدة هى شفرة الأمر والأخرى هى رقم البوابة المراد التعامل معها ، ولذلك فإنه بهذا الأمر يمكن التعامل مع 2^8= 256 بوابة إخراج تبدأ من البوابة رقم 00H وتنتهى بالبوابة رقم FFH . الطريقة الغير مباشرة لهذا الأمر هى :

OUT (C),reg

بوابة الإخراج التى رقمها فى المسجل C ← محتويات المسجل reg

حيث سيقوم هذا الأمر بإخراج المعلومة الموجودة فى المسجل reg الذى يمثل أى مسجل من مسجلات المعالج إلى بوابة الإخراج التى يوجد رقمها فى المسجل C.

مثال 5-6

إفترض أن لدينا خط إنتاج فى أحد المصانع تعبر عليه المنتجات ، وفى أثناء العبور فإن كل منتج يقطع خلية ضوئية فتعطى نبضة كهربية على خرجها . خرج هذه الخلية موصل على البت رقم 0 فى بوابة الإدخال رقم 00H والمطلوب هو كتابة برنامج يعد هذه المنتجات ويخرج العدد على بوابة الإخراج رقم 00H . شكلى (4-11 و 4-12) فى الفصل السابق يبينان رسما توضيحيا ومخطط السير لهذا المثال ، أما البرنامج فتمت إعادته كما فى شكل (5-10) .

5-9 مجموعة أوامر المنطق
Logic Instructions

العمليات المنطقية التى يستطيع المعالج Z80 القيام بها هى العمليات ,XOR NOT, OR, AND وسنكتفى هنا بعرض الصورة العامة لهذه الأوامر على أن يقوم القارىء بمراجعتها فى جداول الأوامر الملحقة فى آخر الفصل . كما ذكرنا سابقا فإن العمليات المنطقية مثلها مثل العمليات الحسابية لابد وأن يكون المسجل A طرفا فيها كما أن النتيجة توضع فى المسجل A .

```
LD B,00; سيكون عداد للمنتج B المسجل
LD C,00; المسجل C يحتوى رقم البوابة التى سنخرج عليها
HERE1: IN A,00 ; A قراءة بوابة الإدخال إلى المسجل
CP 00 ; مقارنة الإشارة بصفر
JP Z,HERE1; طالما أن الإشارة صفر يستمر فى هذه الحلقة
INC B; عند اختلاف الإشارة عن الصفر يزيد B بواحد
HERE2: IN A,00; حلقة انتظار إلى أن ترجع الإشارة للصفر
CP 01
JP Z,HERE2
OUT (C),B ; 00 يخرج محتويات المسجل B على بوابة الإخراج
JP HERE1; يذهب للبداية ليقرأ نبضة جديدة
```

شكل (5-10) برنامج المثال 5-6

جميع العمليات المنطقية تؤثر على الأعلام ما عدا علمي الحمل والحمل النصفي حيث يكونان دائما صفرا بعد أى عملية منطقية لأن الحمل والحمل النصفي غير معرف مع العمليات المنطقية .

الصورة العامة لأوامر العملية AND هى :

```
AND reg
A المسجل ← A المسجل AND reg
AND (HL)
A المسجل ← A المسجل AND (HL)
AND data8
A المسجل ← A المسجل AND data8
AND (IX+d)
A المسجل ← A المسجل AND (IX+d)
AND (IY+d)
A المسجل ← A المسجل AND (IY+d)
```

بنفس الطريقة يمكن كتابة الصورة العامة لأوامر OR و XOR كما يلى :

```
OR reg
OR (HL)
OR data8
OR (IX+d)
```

OR (IY+d)
XOR reg
XOR (HL)
XOR data8
XOR (IX+d)
XOR (IY+d)

هناك عملية NOT وهى لا تجرى إلا على المسجل A حيث يقلب كل صفر إلـــى واحد وكل واحد إلى صفر ، والصورة العامة لهذا الأمر هى :

CPL

عكس المسجل A ⟵ المسجل A

هذه العملية تسمى عملية المتمم الأحادى وهناك عملية المتمم الثنـــائى المعرفـــة كالتالى :

المتمم الثنائى = 1 + المتمم الأحادى

وهناك أمر يقوم بعملية المتمم الثنائى وصورته العامة هى :

NEG

المتمم الثنائى للمسجل A ⟵ المسجل A

إن للمتمم الثنائى أهمية خاصة فى تحويل عمليات الطرح إلى جمـــع كمـــا هـــو مشروح بالتفصيل فى الملحق رقم 1 فى نهاية الكتاب .

إلى هنا نكون قد انتهينا من العرض التفصيلى لمعظم أوامر الشريحة Z80 علـــى أننا سنعرض هذه الأوامر أولا فى صورة مجموعات كما فـــى الأشـــكال (5-11 إلى 5-20) ثم سنعرض الأوامر مرتبة ترتيبا أبجديا كما فى شكل (5-21) .

LD	A,	B,	C,	D,	E,	H,	L,	(HL),	(IX+d),	(IY+d),
A	7F	47	4F	57	5F	67	6F	77	DD77dd	FD77dd
B	78	40	48	50	58	60	68	70	DD70dd	FD70dd
C	79	41	49	51	59	61	69	71	DD71dd	FD71dd
D	7A	42	4A	52	5A	62	6A	72	DD72dd	FD72dd
E	7B	43	4B	53	5B	63	6B	73	DD73dd	FD73dd
H	7C	44	4C	54	5C	64	6C	74	DD74dd	FD74dd
L	7D	45	4D	55	5D	65	6D	75	DD75dd	FD75dd
data8	3E xx	06 xx	0E xx	16 xx	1E xx	26 xx	2E xx	36xx	DD36dd xx	FD36ddxx
(HL)	7E	46	4E	56	5E	66	6E			
(IX+d)	7E	46	4E	56	5E	66	6E			*
(IY+d)	7E	46	4E	56	5E	66	6E			**

* جميع شفرات أوامر هذا الصف تسبقها DD ويعقبها dd مثلها مثل أوامر العمود .,(IX+d)

** جميع شفرات أوامر هذا الصف تسبقها FD ويعقبها dd مثلها مثل أوامر العمود .,(IY+d)

xx يقصد بها البايت الثانية من الأمر وهى data8

LD	BC,	DE,	HL,	SP,	IX,	IY,
(addr)	ED4B adr	ED5B adr	2A adr	ED7B adr	DD2A adr	FD2A adr
data16	01 dat16	11 dat16	21 dat16	31 dat16	DD21 dat16	FD21 dat16

LD (addr),	BC	DE	HL	SP	IX	IY
	ED43 adr	ED53 adr	22 adr	ED73 adr	DD22 adr	FD22 adr

	SR	BC	DE	HL	IX	IY
PUSH	F5	C5	D5	E5	DDE5	FDE5
POP	F1	C1	D1	E1	DDE1	FDE1

LD SP,	HL	IX	IY
	F9	DDF9	FDF9

شكل (5-11) مجموعة أوامر الانتقال للبروسيسور Z80

EX DE,HL	EB
EX (sp),HL	E3
EX (SP),IX	DDE3
EX (SP),IY	FDE3
EX SR,SR1	08
EXX	D9

شكل (5-12) مجموعة أوامر الاستبدال

	ADD A,	ADC A,	SUB	SBC A,	INC	DEC	CP
A	87	8F	97	9F	3C	3D	BF
B	80	88	90	98	04	05	B8
C	81	89	91	99	0C	0D	B9
D	82	8A	92	9A	14	15	BA
E	83	8B	93	9B	1C	1D	BB
H	84	8C	94	9C	24	25	BC
L	85	8D	95	9D	2C	2D	BD
(HL)	86	8E	96	9E	34	35	BE
data8	C6 xx	CE xx	D6 xx	DE xx			FE xx
(IX+d)	DD 86 dd	DD 8E dd	DD 96 dd	DD 9E dd	DD 34 dd	DD 35 dd	DD BE dd
(IY+d)	FD 86 dd	FD 8E dd	FD 96 dd	FD 9E dd	FD 34 dd	FD 35 dd	FD BE dd

شكل (5-13) مجموعة أوامر الحساب

	AND	OR	XOR
A	A7	B7	AF
B	A0	B0	A8
C	A1	B1	A9
D	A2	B2	AA
E	A3	B3	AB
H	A4	B4	AC
L	A5	B5	AD
(HL)	A6	B6	AE
data8	E6xx	F6xx	EExx
(IX+d)	DDA6dd	DDB6dd	DDAEdd
(IY+d)	FDA6dd	FDB6dd	FDAEdd

	CPL	NEG	CCF	SCF
A	2F	ED44	3F	37

شكل (5-14) مجموعة أوامر المنطق

	ADD HL,	ADC HL,	SBC HL,	ADD IX,	ADD IY,	INC	DEC
BC	09	ED 4A	ED 42	DD 09	FD 09	03	0B
DE	19	ED 5A	ED 52	DD 19	FD 19	13	1B
HL	29	ED 6A	ED 62			23	2B
SP	39	ED 7A	ED 72	DD39	FD 39	33	3B
IX				DD 29		DD 23	DD 2B
IY					FD 29	FD 23	FD 2B

شكل (5-15) أوامر حسابية على أزواج مسجلات

JP	JP Z	JP NZ	JP C	JP NC	JP M	JP P	JP PE	JP PO
C3 xx xx	CA xx xx	C2 xx xx	DA xx xx	D2 xx xx	FA xx xx	F2 xx xx	EA xx xx	E2 xx xx

JR	JR Z	JR NZ	JR C	JR NC
18xx	28xx	20xx	38xx	30xx

CALL addr	CDxxxx
CALL Z,addr	CCxxxx
CALL NZ,addr	C4xxxx
CALL C,addr	DCxxxx
CALL NC,addr	D4xxxx
CALL M,addr	FCxxxx
CALL P,addr	F4xxxx
CALL PE,addr	ECxxxx
CALL PO,addr	E4xxxx

RET	C9
RET Z	C8
RET NZ	C0
RET C	D8
RET NC	D0
RET M	F8
RET P	F0
RET PE	E8
RET PO	E0

xxxx تمثل اثنين بايت للعنوان الذى سيتم القفز إليه

xx تمثل بايت واحدة للعنوان الذى سيتم القفز إليه فى حالة القفز النسبى

شكل (5-16) مجموعة أوامر القفز

الأمر BIT b,sss يجعل علم الصفر يساوى عكس البت رقم b فى المسجل أو الذاكرة sss

BIT	0,	1,	2,	3,	4,	5,	6,	7,
A	CB 47	CB 4F	CB 57	CB 5F	CB 67	CB 6F	CB 77	CB 7F
B	CB 40	CB 48	CB 50	CB 58	CB 60	CB 68	CB 70	CB 78
C	CB 41	CB 49	CB 51	CB 59	CB 61	CB 69	CB 71	CB 79
D	CB 42	CB 4A	CB 52	CB 5A	CB 62	CB 6A	CB 72	CB 7A
E	CB 43	CB 4B	CB 53	CB 5B	CB 63	CB 6B	CB 73	CB 7B
H	CB 44	CB 4C	CB 54	CB 5C	CB 64	CB 6C	CB 74	CB 7C
L	CB 45	CB 4D	CB 55	CB 5D	CB 65	CB 6D	CB 75	CB 7D
(HL)	CB 46	CB 4E	CB 56	CB 5E	CB 66	CB 6E	CB 76	CB 7E
(IX+d)	DD CB d46	DD CB d4E	DD CB d56	DD CB d5E	DD CB d66	DD CB d6E	DD CB d76	DD CB d7E
(IY+d)	FD CB d46	FD CB d4E	FD CB d56	FD CB d5E	FD CB d66	FD CB d6E	FD CB d76	FD CB d7E

شكل (5-17) مجموعة اختبار و SET و RESET بت من بتات مسجل أو مكان فى الذاكرة

الأمر SET b,sss يجعل البت رقم b فى المسجل أو الذاكرة sss تساوى واحد

SET	0,	1,	2,	3,	4,	5,	6,	7,
A	CB C7	CB CF	CB D7	CB DF	CB E7	CB EF	CB F7	CB FF
B	CB C0	CB C8	CB D0	CB D8	CB E0	CB E8	CB F0	CB F8
C	CB C1	CB C9	CB D1	CB D9	CB E1	CB E9	CB F1	CB F9
D	CB C2	CB CA	CB D2	CB DA	CB E2	CB EA	CB F2	CB FA
E	CB C3	CB CB	CB D3	CB DB	CB E3	CB EB	CB F3	CB FB
H	CB C4	CB CC	CB D4	CB DC	CB E4	CB EC	CB F4	CB FC
L	CB C5	CB CD	CB D5	CB DD	CB E5	CB ED	CB F5	CB FD
(HL)	CB C6	CB CE	CB D6	CB DE	CB E6	CB EE	CB F6	CB FE
(IX+d)	DD CB dC6	DD CB dCE	DD CB dD6	DD CB dDE	DD CB dE6	DD CB dEE	DD CB dF6	DD CB dFE
(IY+d)	FD CB dC6	FD CB dCE	FD CB dD6	FD CB dDE	FD CB dE6	FD CB dEE	FD CB dF6	FD CB dFE

تابع شكل (5-17) مجموعة اختبار و SET و RESET بت من بتات مسجل أو مكان فى الذاكرة

الأمر RES b,sss يجعل البت رقم b فى المسجل أو الذاكرة sss تساوى صفر

RES	0,	1,	2,	3,	4,	5,	6,	7,
A	CB 87	CB 8F	CB 97	CB 9F	CB A7	CB AF	CB B7	CB BF
B	CB 80	CB 88	CB 90	CB 98	CB A0	CB A8	CB B0	CB B8
C	CB 81	CB 89	CB 91	CB 99	CB A1	CB A9	CB B1	CB B9
D	CB 82	CB 8A	CB 92	CB 9A	CB A2	CB AA	CB B2	CB BA
E	CB 83	CB 8B	CB 93	CB 9B	CB A3	CB AB	CB B3	CB BB
H	CB 84	CB 8C	CB 94	CB 9C	CB A4	CB AC	CB B4	CB BC
L	CB 85	CB 8D	CB 95	CB 9D	CB A5	CB AD	CB B5	CB BD
(HL)	CB 86	CB 8E	CB 96	CB 9E	CB A6	CB AE	CB B6	CB BE
(IX+d)	DD CB d86	DD CB d8E	DD CB d96	DD CB d9E	DD CB dA6	DD CB dAE	DD CB dB6	DD CB dBE
(IY+d)	FD CB d86	FD CB d8E	FD CB d96	FD CB d9E	FD CB dA6	FD CB dAE	FD CB dB6	FD CB dBE

تابع شكل (5-17) مجموعة اختبار و SET و RESET بت من بتات مسجل أو مكان فى الذاكرة

	RLC	RRC	RL	RR	SLA	SRA	SRL
A	CB 07	CB 0F	CB 17	CB 1F	CB 27	CB 2F	CB 3F
B	CB 00	CB 08	CB 10	CB 18	CB 20	CB 28	CB 38
C	CB 01	CB 09	CB 11	CB 19	CB 21	CB 29	CB 39
D	CB 02	CB 0A	CB 12	CB 1A	CB 22	CB 2A	CB 3A
E	CB 03	CB 0B	CB 13	CB 1B	CB 23	CB 2B	CB 3B
H	CB 04	CB 0C	CB 14	CB 1C	CB 24	CB 2C	CB 3C
L	CB 05	CB 0D	CB 15	CB 1D	CB 25	CB 2D	CB 3D
(HL)	CB 06	CB 0E	CB 16	CB 1E	CB 26	CB 2E	CB 3E
(IX+d)	DD CB d06	DD CB d0E	DD CB d16	DD CB d1E	DD CB d26	DD CB d2E	DD CB d3E
(IY+d)	FD CB d06	FD CB d0E	FD CB d16	FD CB d1E	FD CB d26	FD CB d2E	FD CB d3E

أوامر خاصة بإزاحة أو دوران المسجل A فقط

RRA	RLA	RRCA	RLCA
1F	17	0F	07

شكل (5-18) مجموعة أوامر الأزاحة والدوران

IN	A,	B,	C,	D,	E,	H,	L,
Port No.	D8xx	-----	-----	-----	-----	-----	-----
(C)	ED78	ED40	ED48	ED50	ED58	ED60	ED68

OUT	Port No.,	(C),
A	D3xx	ED79
B	-----	ED41
C	-----	ED49
D	-----	ED51
E	-----	ED59
H	-----	ED61
L	-----	ED69

شكل (5-19) أوامر الإدخال والإخراج

No Operation, NOP لا تعمل شيء	00
HALT توقف	76
Disable Interrupt, DI إخماد المقاطعة	F3
Enable Interrupt, EI تنشيط المقاطعة	FB
IM0 تنشيط حالة المقاطعة رقم صفر	ED46
IM1 تنشيط حالة المقاطعة رقم صفر	ED56
IM2 تنشيط حالة المقاطعة رقم صفر	ED5E

شكل (5-20) مجموعة أوامر متفرقة

كانت هذه بعض أهم مجموعات الأوامر للمعالج Z80 والشائعة الاستخدام . شكل (5-21) يحتوى جميع أوامر الشريحة مرتبة ترتيبا أبجديا مع نبذة عن ما يعمله كل أمر وعدد نبضات التزامن التى يأخذها (ن #) لكى يتم إحضاره من الذاكرة وتنفيذه والأعلام التى تتأثر بكل أمر وكذلك شفرة كل أمر حيث من هذه الشفرة يمكن استنتاج عدد بايتات الأمر . انظر الملاحظات الخاصة بشفرة الأوامر فى نهاية هذا الشكل .

شفرة الأسمبلى	الأعلام المتأثرة	# ن	شفرة الأمر	وظيفة الأمر
ADC A,reg	ZSP CY HC	4	10001sss	A ← A+CY+ reg
ADC A,(HL)	ZSP CY HC	7	8E	A ← A+CY+(HL)
ADC A,data8	ZSP CY HC	7	CE data8	A ← A+CY+data8
ADC A,(IY+d)	ZSP CY HC	19	FD 8E dd	A ← A+CY+(IY+d)
ADC A,(IX+d)	ZSP CY HC	19	DD 8E dd	A ← A+CY+(IX+d)
ADC HL,rp	- - - CY -	15	ED 01rp1010	HL ← HL+CY+rp
ADD A,reg	ZSP CY HC	4	10000sss	A ← A + Reg
ADD A,(HL)	ZSP CY HC	7	86	A ← A + (HL)
ADD A,data8	ZSP CY HC	7	C6 data8	A ← A+data8
ADD A,(IY+d)	ZSP CY HC	19	FD 88 dd	A ← A+ (IY+d)
ADD A,(IX+d)	ZSP CY HC	19	DD 86 dd	A ← A+ (IX+d)
ADD HL,rp	- - - CY -	11	00rp1001	HL ← HL+rp
ADD IY,rp	- - - CY -	15	FD 00rp1001	IX ← IY + rp
ADD IX,rp	- - - CY -	15	DD 00rp1001	IX ← IX + rp
AND reg	ZS P 0 0	4	10100xxx	A ← A AND reg
AND (HL)	ZS P 0 0	7	A6	A ← A AND (HL)
AND data8	ZS P 0 0	7	E6 data8	A ← A AND data8
AND (IY+d)	ZS P 0 0	19	FD A6 dd	A ← A AND (IY+d)
AND (IX+d)	ZS P 0 0	19	DD A6 dd	A ← A AND (IX+d)
BIT b,reg	Z - - - -	9	CB 01bbbsss	ZF ← عكس البت b فى reg
BIT b,(HL)	Z - - - -	12	CB 01bbb110	ZF ← عكس البت b فى (HL)
BIT b,(IY+d)	Z - - - -	20	FD CB dd 01bbb110	ZF ← عكس البت b فى (IY+D)
BIT b,(IX+d)	Z - - - -	20	DD CB dd 01bbb110	ZF ← عكس البت b فى (IX+D)
CALL addr	- - - - - - -	17	CD addr	نداء غير مشروط لبرنامج فرعى
CALL C,addr	- - - - - - -	10/17	DC addr	نداء مشروط بعلم الحمل=1
CALL M,addr	- - - - - - -	10/17	FC addr	نداء مشروط بعلم إشارة=1
CALL NC,addr	- - - - - - -	10/17	D4 addr	نداء مشروط بعلم الحمل=0
CALL NZ,addr	- - - - - - -	10/17	C4 addr	نداء مشروط بعلم الصفر=0
CALL P,addr	- - - - - - -	10/17	F4 addr	نداء مشروط بعلم إشارة=0
CALL PE,addr	- - - - - - -	10/17	EC addr	نداء مشروط بعلم باريتى=1
CALL PO,addr	- - - - - - -	10/17	E4 addr	نداء مشروط بعلم باريتى=0
CALL Z,addr	- - - - - - -	10/17	CC addr	نداء مشروط بعلم الصفر=1
CCF	- - - CY HC	4	3F	اعكس علم الحمل
CP reg	ZSP CY HC	4	10111sss	مقارنة A − reg
CP (HL)	ZSP CY HC	7	BE	مقارنة A − (HL)
CP const	ZSP CY HC	7	FE data8	مقارنة A - const
CP (IY+d)	ZSP CY HC	19	FD BE data8	مقارنة A - (IY+d)
CP (IX+d)	ZSP CY HC	19	DD BE data8	مقارنة A - (IX+d)
CPL	- - - - - -	4	2F	اعكس المسجل A

CPI	ZSP CY HC	16	ED A1	مقارنة A-(HL) BC← BC-1 ,HL← HL+
CPIR	ZSP CY HC	21/16	ED B1	كرر CPI إلى BC=0
CPD	ZSP CY HC	16	ED A9	مقارنة A-(HL) BC← BC-1 ,HL←HL-
CPDR	ZSP CY HC	21/16	ED B9	كرر CPD إلى BC=0
DAA	ZSP CY HC	4	27	حول المركم للنظام العشرى
DEC reg	ZSP -- HC	4	00ddd101	reg ← reg -
DEC (HL)	ZSP -- HC	7	35	(HL) ← (HL) - 1
DEC (IY+d)	ZSP -- HC	19	FD 35 data8	(IX+d) ← (IY+d) -
DEC (IX+d)	ZSP -- HC	19	DD 35 data8	(IX+d) ← (IX+d) -
DI	------	4	F3	أهمل المقاطعة IF ← 0
DEC rp	------	6	00rp1011	rp ← rp - 1
DEC IY	------	10	FD 2B	IX ← IY - 1
DEC IX	------	10	DD 2B	IX ← IX - 1
DJNZ ddd	------	8/13	10 ddd	قفز بمقدار ddd وإنقاص B بمقدار 1 إلى أن يصبح B=0
EI	------	4	FB	اسمح بالمقاطعة IF← 1
EX DE,HL	------	4	EB	HL ↔ DE
EX AF,AF1	------	4	08	PSW1 ↔ PSW
EXX	------	4	D9	BCDEHL1 ↔ BCDEHL
EX (SP),HL	------	19	E3	(SP+1) → H (SP) → L
EX (SP),IX	------	23	FD E3	(SP+1) → IY/H (SP) → IY/L
EX (SP),IX	------	23	DD E3	(SP+1) → IX/H (SP) → IX/L
HALT		4	76	أوقف تنفيذ البرنامج
IM 0	------	8	ED 46	حالة المقاطعة صفر
IM 1	------	8	ED 56	حالة المقاطعة واحد
IM 2	------	8	ED 5E	حالة المقاطعة اثنين
IN A,(no.)	------	11	DB no8	A ← البوابة رقم .no
IN reg,(C)	------	12	ED 01ddd000	reg ← البوابة (C)
INI	------	16	ED A2	port (C) → (HL) B-1 → B HL ← HL+1
INIR	------	16/21	ED B2	كرر INI إلى أن B=0
IND	------	16	ED AA	port(C) → (HL) B-1 → B HL ← HL-1
INDR	------	16/21	ED BA	كرر IND إلى أن B=0
INC reg	Z S P - HC	4	00ddd100	reg ← reg + 1
INC (HL)	Z S P - HC	7	34	(HL) + 1 → (HL)
INC (IY+d)	Z S P - HC	19	FD 34 dd	(IY+d) ← (IY+d) + 1
INC (IX+d)	Z S P - HC	19	DD 34 dd	(IX+d) ← (IX+d) + 1

INC rp	------	6	00xx0011	rp ← rp + 1
INC IY	------	10	FD 23	IY ← IY + 1
INC IX	------	10	DD 23	IX ← IX + 1
JP addr	------	10	C3 addr	قفز غير مشروط
JP (HL)	------	4	E9	قفز إلي عنوان في HL
JP (IX)	------	8	DD E9	قفز إلي عنوان في IX
JP (IY)	------	8	FD E9	قفز إلي عنوان في IY
JP Z,addr	------	10	CA addr	قفز مشروط بعلم الصفر=1
JP NZ,addr	------	10	C2 addr	قفز مشروط بعلم الصفر=0
JP C,addr	------	10	DA addr	قفز مشروط بعلم الحمل=1
JP NC,addr	------	10	D2 addr	قفز مشروط بعلم الحمل=0
JP PO,addr	------	10	E2 addr	قفز مشروط بعلم باريتي=0
JP PE,addr	------	10	EA addr	قفز مشروط بعلم باريتي=1
JP P,addr	------	10	F2 addr	قفز مشروط بعلم اشارة=0
JP M,addr	------	10	FA addr	قفز مشروط بعلم اشارة=1
JR dd	------	10	18 dd	قفز نسبي غير مشروط
JR Z,dd	------	7/12	28 dd	قفز نسبي مشروط Z=1
JR NZ,dd	------	7/12	20 dd	قفز نسبي مشروط Z=0
JR C,dd	------	7/12	38 dd	قفز نسبي مشروطCY=1
JR NC,dd	------	7/12	30 dd	قفز نسبي مشروطCY=0
LD reg1,reg2	------	4	01dddsss	reg2 → reg1
LD reg,(HL)	------	7	01ddd110	(HL) → reg
LD (HL),reg	------	7	01110sss	(HL) ← reg
LD reg,data8	------	7	11ddd110 data8	reg ← data8
LD reg,(IY+d)	------	19	FD 01ddd110 dd	reg ← (IY+d)
LD reg,(IX+d)	------	19	DD 01ddd110 dd	reg ← (IX+d)
LD (IY+d),reg	------	19	FD 01110sss dd	(IY+d) ← reg
LD (IX+d),reg	------	19	DD 01110sss dd	(IX+d) ← reg
LD (HL),data8	------	10	36 data8	(HL) ← data8
LD (IY+d),data8	------	19	FD 36 dd data8	(IY+d) ← data8
LD (IX+d),data8	------	19	DD 36 dd data8	(IX+d) ← data8
LD A,(addr)	------	13	3A addr	A ← (addr)
LD (addr),A	------	13	32 addr	(addr) ← A
LD (addr),BC	------	20	ED 43 addr	addr ← C addr+1 ← B
LD (addr),DE	------	20	ED 53 addr	addr ← E addr+1 ← D
LD (addr),HL	------	20	22 addr	addr ← L addr+1 ← H
LD (addr),IX	------	20	DD 22 addr	addr ← IX/L addr+1 ← IX/H
LD (addr),IY	------	20	FD 22 addr	addr ← IY/L addr+1 ← IY/H
LD (addr),SP	------	20	ED 73 addr	addr ← SP/L addr+1 ← SP/H
LD A,(BC)	------	7	0A	A ← (BC)
LD A,(DE)	------	7	1A	A ← (DE)
LD (BC),A	------	7	02	A → (BC)
LD (DE),A	------	7	12	A → (DE)
LD A,I	------	9	ED 57	A ← I
LD I,A	------	9	ED 47	I ← A

LD A,R	------	9	ED 5F	A ← R
LD R,A	------	9	ED 4F	R ← A
LD rp,data16	------	10	00rp0001 data16	rp ← data16
LD IX,data16	------	14	DD 21 data16	IX ← data16
LD rp,(addr)	------	20	ED 01rp1011 addr	(addr+1) → B (addr) → C
LD HL,(addr)	------	16	2A addr	(addr+1) → H (addr) → L
LD IX,(addr)	------	20	DD 2A addr	(addr) → IX/L IX/H ← (addr+1),
LD IY,(addr)	------	20	FD 2A addr	(addr) → IY/L IY/H ← (addr+1),
LD SP,HL	------	16	F9	HL → SP
LD SP,IX	------	10	DD F9	SP ← IX
LD SP,IY	------	10	FD F9	SP ← IY
LDI	Z S P - -	16	ED A0	DE ← DE+1 (HL) → (DE) HL ← HL+1 BC ← BC-1
LDIR	Z S P - -	21/16	ED B0	نفس الأمر السابق إلى BC=0
LDD	Z S P - -	16	ED AB	DE ← DE-1 (HL) → (DE BC ← BC-1 HL ← HL-1
LDDR	Z S P - -	21/16	ED BB	نفس الأمر السابق إلى BC=0
NEG	Z S P - -	8	ED 44	المتمم الثنائي ل A ← A
NOP	------	4	00	لا تعمل شيئا No operation
OR reg	Z S P 0 0	4	10110sss	A ← A OR reg
OR (HL)	Z S P 0 0	7	B5	A ← A OR (HL)
OR data8	Z S P 0 0	7	F6 data8	A ← A OR data8
OR (IY+d)	Z S P 0 0	19	FD B6 dd	A ← A OR (IY+d)
OR (IX+d)	Z S P 0 0	19	DD B6 dd	A ← A OR (IX+d)
OUT (no.),A	------	11	D3 no.8	port (no.) ← A
OUT (C),reg	------	12	ED 01sss001	port (C) ← reg
OUTI	------	16	ED A3	(HL) → port(C) B-1 → B HL → HL+1
OTIR	------	16/21	ED B3	كرر OUTI إلى أن B=0
OUTD	------	16	ED A8	B ← B-1,port (C) ← (HL) HL ← HL-1
OTDR	------	16/21	ED B8	كرر OUTD إلى أن B=0
PUSH rp	------	11	11rp0101	المسجلان rp ← قمة المكدسة
PUSH IY	------	15	FD E5	المسجل IY ← قمة المكدسة
PUSH IX	------	15	DD E5	المسجل IX ← قمة المكدسة
POP rp	------	11	11rp0001	قمة المكدسة ← المسجلين rp
POP IY	------	15	FD E1	قمة المكدسة ← المسجل IX
POP IX	------	15	DD E1	قمة المكدسة ← المسجل IX
RLCA	- - - CY -	4	07	CY ← A المسجل ←

RLA	--- CY -	4	17	CY ← A المسجل ←
RRCA	--- CY -	4	0F	A المسجل → CY
RRA	--- CY -	4	1F	A المسجل → CY
RLC reg	--- CY -	8	CB 0000sss	دوران مسجل , مثل RLCA
RLC (HL)	--- CY -	15	CB 06	دوران عنوان فى HLمثل RLCA
RLC (IY+d)	--- CY -	23	FD CB dd 06	دوران عنوان(IY+d)مثل RLCA
RLC (IX+d)	--- CY -	23	DD CB dd 06	دوران عنوان(IX+d)مثل RLCA
RL (HL)	--- CY -	15	CB 16	دوران (HL) لليسار من خلال علم الحمل
RL (IX+d)	--- CY -	23	DD CB dd 16	دوران (IX+d) لليسار من خلال علم الحمل
RL (IY+d)	--- CY -	23	FD CB dd 16	دوران (IY+d) لليسار من خلال علم الحمل
RL reg	--- CY -	8	CB 00010sss	دوران reg لليسار من خلال علم الحمل
RLD	------	18	ED 6F	دوران 4بت الأولى من A للشمال مع العنوان الموجود فى HL
RRD	------	18	ED 67	دوران 4بت الأولى من A لليمين مع العنوان الموجود فى HL
RR (HL)	--- CY -	15	CB 1E	دوران (HL) لليمين من خلال علم الحمل
RR (IX+d)	--- CY -	23	DD CB dd 1E	دوران (IX+d) لليمين من خلال علم الحمل
RR (IY+d)	--- CY -	23	FD CB dd 1E	دوران (IY+d) لليمين من خلال علم الحمل
RR reg	--- CY -	8	CB 00011sss	دوران reg لليمين من خلال علم الحمل
RRC (HL)	--- CY -	15	CB 0E	دوران (HL) لليمين مثل RRCA
RRC (IX+d)	--- CY -	23	DD CB dd 0E	دوران (IX+d) لليمين مثل RRCA
RRC (IY+d)	--- CY -	23	FD CB dd 0E	دوران (IY+d) لليمين مثل RRCA
RRC reg	--- CY -	8	CB 00001sss	دوران reg لليمين مثل RRCA
RET	------	10	C9	عودة من برنامج فرعى
RET Z	------	5/11	C8	عودة مشروطة بعلم الصفر=1
RET NZ	------	5/11	C0	عودة مشروطة بعلم الصفر=0
RET C	------	5/11	D8	عودة مشروطة بعلم الحمل=1
RET NC	------	5/11	D0	عودة مشروطة بعلم الحمل=0
RET PO	------	5/11	E0	عودة مشروطة بعلم باريتى=0
RET PE	------	5/11	E8	عودة مشروطة بعلم باريتى=1
RET M	------	5/11	F8	عودة مشروطة بعلماشارة=1
RET P	------	5/11	F0	عودة مشروطة بعلم إشارة=0
RETI	------	14	ED 4D	عودة من مقاطعة
RETN	------	14	ED 45	عودة من مقاطعة ذات قناع

RES b,reg	- - - - - -	8	CB 10bbbsss	صفر فى البت b فى reg
RES b,(HL)	- - - - - -	15	CB 10bbb110	صفر فى البت b فى (HL)
RES b,(IX+d)	- - - - - -	23	DD CB dd 10bbb110	صفر فى البت b فى (IX+d)
RES b,(IY+d)	- - - - - -	23	FD CB dd 10bbb110	صفر فى البت b فى (IY+d)
RST n	- - - - - -	11	11nnn111	إعادة تشغيل
SUB reg	Z S P CY HC	4	10010sss	A ⟵ A - reg
SUB (HL)	Z S P CYHC	7	96	A ⟵ A - (HL)
SUB data8	Z S P CY HC	7	D6 data8	A ⟵ A - data8
SUB (IX+d)	Z S P CY HC	19	DD 96 dd	A ⟵ A - (IX+d)
SUB (IY+d)	Z S P CY HC	19	FD 96 dd	A ⟵ A - (IY+d)
SBC reg	Z S P CY HC	4	10011sss	A ⟵ A - CY – reg
SBC (HL)	Z S P CY HC	7	9E	A ⟵ A - CY - (HL)
SBC data8	Z S P CY HC	7	DE data8	A ⟵ A - CY - data8
SBC (IX+d)	Z S P CY HC	19	DD 9E dd	A ⟵ A - CY - (IX+d)
SBC (IY+d)	Z S P CY HC	19	FD 9E dd	A ⟵ A - CY - (IY+d)
SBC HL,rp	Z S P CY HC	15	ED 01rp0010	HL ⟵ HL - CY – rp
SLA (HL)	Z S P CY HC	2	CB 26	إزاحة (HL) لليسار ، حمل 0 فى بت 0
SLA (IX+d)	Z S P CY HC	4	DD CB dd 26	إزاحة (IX+d) لليسار ، حمل 0 فى بت 0
SLA (IY+d)	Z S P CY HC	4	FD CB dd 26	إزاحة (IY+d) لليسار ، حمل 0 فى بت 0
SLA reg	Z S P CY HC	2	CB 00100sss	إزاحة reg لليسارحمل 0 فى بت0
SRL (HL)	Z S P CY HC	2	CB 3E	إزاحة (HL) لليمين ، حمل 0 فى بت 7
SRL (IX+d)	Z S P CY HC	4	DD CB dd 3E	إزاحة (HL) لليمين ، حمل 0 فى بت 7
SRL (IY+d)	Z S P CY HC	4	FD CB dd 3E	إزاحة (HL) لليمين ، حمل 0 فى بت 7
SRL reg	Z S P CY HC	2	CB 00111sss	إزاحة reg لليمين ، حمل 0 فى بت 7
SRA (HL)	Z S P CY HC	2	CB 2E	إزاحة (HL) لليمين ، بت 7 تظل كما هى
SRA (IX+d)	Z S P CY HC	4	DD CB dd 2E	إزاحة (IX+d) لليمين ، بت 7 تظل كما هي
SRA (IY+d)	Z S P CY HC	4	FD CB dd 2E	إزاحة (IY+d) لليمين ، بت 7 تظل كما هي
SRA reg	Z S P CY HC	2	CB 00101sss	إزاحة reg لليمين،بت7تظل كماهى
SET b,reg	Z S P CY HC	8	CB 11bbbsss	واحد فى البت b فى reg
SET b,(IY+d)	Z S P CY HC	23	FD CB dd 11bbb110	واحد فى البت b فى (IY+d)
SET b,(HL)	Z S P CY HC	15	DD CB dd 11bbb110	واحد فى البت b فى (HL)

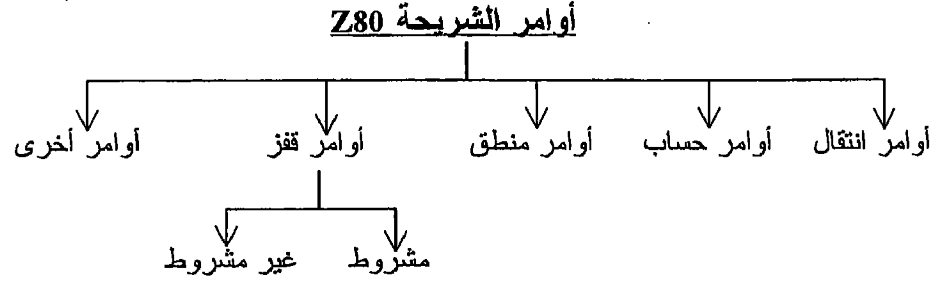
SET b,(IX+d)	Z S P CY HC	23	DD CB dd 11bbb110	واحد فى البت b فى (IX+d)
XOR reg	Z S P 0 0	4	10101sss	A ⟵ A XOR reg
XOR (HL)	Z S P 0 0	7	AE	A ⟵ A XOR (HL)
XOR data8	Z S P 0 0	7	EE data8	A ⟵ A XOR data8
XOR (IY+d)	Z S P 0 0	19	FD AE dd	A ⟵ A XOR (IY+d)
XOR (IX+d)	Z S P 0 0	19	DD AE dd	A ⟵ A XOR (IX+d)

شكل (5-21) أوامر الشريحة Z80 مرتبة أبجديا

- sss أو ddd تستبدل بشفرة مسجل 8 بت كما فى فصل 2 .
- rp تستبدل بشفرة زوج مسجلات كما فى فصل 2 .
- data8 ثابت مكون من 8 بت (بايت) .
- dd إزاحة عنوانية مكونة من 8 بت (بايت) .
- bbb رقم بت معينة فى مسجل أو بايت ذاكرة .
- no.8 رقم (عنوان) بوابة (إدخال أو إخراج) من 8 بت (بايت) .

5-10 تمارين

1. أكمل الجدول التالى الخاص بأوامر الشريحة Z80 :



2. ما هى نتيجة تنفيذ البرنامج التالى :

```
E000  LD A,05
E002  LD B,A
E003  LD C,B
E004  LD D,C
E005  LD E,D
E006  LD L,E
E007  LD H,L
E008  LD (HL),L
```

3. اقرأ البرنامج السابق وأجب عما يلى :
- محتويات مكان الذاكرة E000=..........

- محتويات مكان الذاكرة ....=E001
- محتويات مكان الذاكرة ....=0505

.4

```
E000 LD HL,E100
E003 LD (HL),3E
E005 INC HL
E006 LD (HL),05
E008 INC HL
E009 LD (HL),47
E00B INC HL
E00C LD (HL),48
```

ما هى نتيجة تنفيذ البرنامج السابق ؟

5. على ضوء نتيجة تنفيذ البرنامج السابق ما هى نتيجة تنفيذ الشفرات الموجـودة فى الأماكن E100 إلى E103 ؟
6. هل تتأثر الأعلام بأوامر الانتقال ؟
7. أذكر الأعلام التى تتأثر بكل عملية من العمليات الحسابية والمنطقية؟
8. إذا كانت محتويات المسجل A=F3H ومحتويات المسـجل B=A4H فـاكتب محتويات المسجل A بعد تنفيذ كل أمر من الأوامر التالية على نفس المحتويـات السابقة ووضح أيضا كيف ستتأثر الأعلام بكل أمر :

```
ADD A,B
SUB A,B
SUB A,A
INC A
AND B
OR B
XOR B
```

9. ارسم مخطط السير للبرنامج التالى وما هى نتيجة تنفيذه :

```
E000 LD L,50H
E003  LD H,E1H
E005  LD (HL),A
E006  DEC L
E007  JPNZ E005
```

10. ماذا يحدث لو كتبنا البرنامج السابق عند E100 بدلا من E000 ؟
11. أعد كتابة البرنامج السابق مستخدما العلامات Labels ؟ وما هـى مميـزات البرنامج مكتوبا بهذه الصورة ؟
12. اكتب برنامجا يحسب عدد الوحايد الموجودة فى محتويات المسجل A ، مثـلا إذا كان A=1110101 فإن عدد الوحايد = 6 .

13. كم عدد بوابات الإدخال التى يستطيع البروسيسور Z80 التعامل معها؟
14. كم عدد بوابات الإخراج التى يستطيع البروسيسور Z80 التعامل معها؟
15. على ماذا يتوقف هذا العدد ؟
16. هل هناك ما يمنع أن تكون بوابتى إدخال وإخراج لهما نفس الرقم , كمثال على ذلكIN 05 و OUT 05؟
17. OUT (C),reg هذا أحد أوامر الإخراج للبروسيسور Z80 والذى يعنى أخراج محتويات المسجل reg على بوابة الإخراج التى رقمها فى المسجل C فهل البروسيسور 8085 لديه ما يكافىء هذا الأمر ؟
18. هل تتأثر الأعلام بأوامر الإدخال والإخراج ؟
19. أكتب برنامجا يقرأ محتويات البوابة 00 وإذا كانت هذه المحتويات زوجية يخزنها فى الذاكرة ابتداء من العنوان E100 وإذا كانت فردية يخرجها على البوابة 00 ؟
20. اذكر طرق العنونة memory addressings المستخدمة مع البروسيسور Z80؟ والأوامر المستخدمة مع كل طريقة ؟ ومتى يفضل استخدام كل طريقة ؟
21. اكتب برنامج يحسب أكبر قيمة عددية فى بايت فى المدى العنوانى E200H إلى E250H .
22. اكتب برنامج يحسب عدد البايتات التى تحتوى أصفرا والتى تحتوى أرقاما موجبة والتى تحتوى أرقاما سالبة فى المدى العنوانى E100 إلى E150 .
23. اكتب برنامج يحسب عدد البايتات التى تحتوى بيانات فردية والتى تحتوى بيانات زوجية فى المدى العنوانى E100 إلى E150 .
24. المدى العنوانى E100 إلى E150 يحتوى بيانات لإشارة صوت ، احسب كم مرة عبرت إشارة الصوت الصفر .
25. اكتب برنامج يقرأ بوابة الإدخال رقم 00 ويختبر البت الرابعة فيها ، فإذا كانت هذه البت واحد يخرج هذه المحتويات على البوابة 00 ، وإذا كانت هذه البت صفر يخرج محتوياتها على البوابة 01 .
26. اكتب برنامج يقرأ بوابة الإدخال رقم 00 إلى مالانهاية ويختبر البيانات التى يقرأها ، فإن كانت فردية يخرجها على البوابة 00 ، وإن كانت زوجية يخرجها على البوابة 01 . احسب أكبر معدل لدخول البيانات لكى يعمل هذا النظام فى الزمن المباشر real time .

الفصل السادس

المعالج

من البداية ... حتى النهاية

Microprocessor... from Start ... to end

6-1 مقدمة

سنقوم فى هذا الفصل بعملية بناء تدريجية لمعالج افتراضى يقوم بعدد محدود من العمليات الحسابية والمنطقية وله عدد محدود من الأوامر كما أن له عددا محدودا جدا من الخطوط فى مسارات البيانات والعناوين والتحكم ولذلك فإن هذا المعالج يستطيع التعامل مع كمية محدودة جدا من بايتات الذاكرة وسنبدأ عملية البناء من أقل مستوى ممكن ثم سنرتقى بها خطوة بخطوة إلى أن نصل إلى معالج متكامل ولكن بالمواصفات التى ذكرناها سابقا . من خلال عملية البناء سنتعرف على وحدة الحساب والمنطق وكيفية عملها . ولقد كان قصدنا من وضع هذا الفصل فى هذا الترتيب أن يكون القارىء قد ألم بفكرة عامة عن تركيب المعالج من الفصول السابقة ثم يجىء هذا الفصل فيؤكد هذه الفكرة ويمحصها ويضيف إليها التفاصيل الدقيقة التى قد تجيب على الكثير من الأسئلة التى تدور فى خلد أى قارىء عن كيفية تنفيذ أى معالج لأى أمر وما الذى يتحكم فى عدد أوامره وغير ذلك من الأسئلة المهمة .

إن وحدة الحساب والمنطق هى إحدى المكونات الرئيسية للمعالج ومهمتها الأساسية هى إجراء العمليات الحسابية والمنطقية الأساسية وسنبدأ فيما يلى عملية بناء هذه الوحدة ولكى نصل إلى ذلك لابد أن نتعرف أولا على كيفية تنفيذ عمليات الجمع والطرح فى النظام الثنائى .

6-2 الجمع الثنائى Binary addition

مثال 6-1

أوجد ناتج جمع الرقم A=a3a2a1a0 =1101 مع الرقم B=b3b2b1b0 =1011.

إن عملية الجمع تتم كالتالى:

```
الحمل  1  1  1  1
   A      1  1  0  1
   B      1  0  1  1  +
حمل←1     1  0  0  0
        S3 S2 S1 S0  النتيجة
```

كما نرى فإن عملية الجمع تتم على عدد من المراحل ، المرحلة الأولى هى جمع a0 (البت الأولى فى الرقم A) مع b0 (البت الأولى من الرقم B) فينتج من ذلك نتيجة s0=0 وحمل c0=1 إلى المرحلة التالية . فى المرحلة الثانية يتم جمع البتات الآتية :

$c0 + b1 + a1$

حيث c0 هى الحمل من المرحلة السابقة كما ذكرنا . نتيجة جمع المرحلة الثانية ستكون s1=0 والحمل منها سيكون c1=1 إلى المرحلة التالية . فى المرحلة الثالثة ستتم عملية الجمع التالية :

$c1 + b2 + a2$

وسينتج عنها s2=0 و c2=1 ثم فى مرحلة الجمع الرابعة سيتم جمع البتات التالية:

$c2 + b3 + a3$

وسينتج عنها s3=1 و c3=1 وبذلك تنتهى عملية الجمع ويتبقى حمل أخير للخانة الرابعة وهو c3=1 الذى سنهمله الآن . من ذلك نرى أننا فى المرحلة الأولى نجمع اثنين بت فقط (a0 + b0) وأما فى باقى المراحل فإننا نجمع ثلاث بتات (an + bn + cn-1) حيث an هى البت رقم n فى العدد A و bn هى البت رقم n أيضا فى العدد B ، وأما cn-1 فهى الحمل الناتج من جمع المرحلة السابقة (cn-2 +bn-1+an-1) . الآن نريد تكوين دائرة منطقية تقوم بعملية جمع 2 بت وأخرى تقوم بعملية جمع 3 بتات ثم من الدائرتين نقوم بتكوين دائرة تجمع العددين A و B .

6-2-1 دائرة نصف المجمع Half adder circuit, HA

يقوم نصف المجمع Half Adder, HA بجمع اثنين بت a0 و b0 ويعطى فى الخرج النتيجة s0 وحمل c0 ويبين شكل (6-1) جدول الحقيقة truth table لهذه الدائرة . من جدول الحقيقة نستطيع كتابة المعادلات المنطقية التالية لكل من خرجى دائرة نصف المجمع :

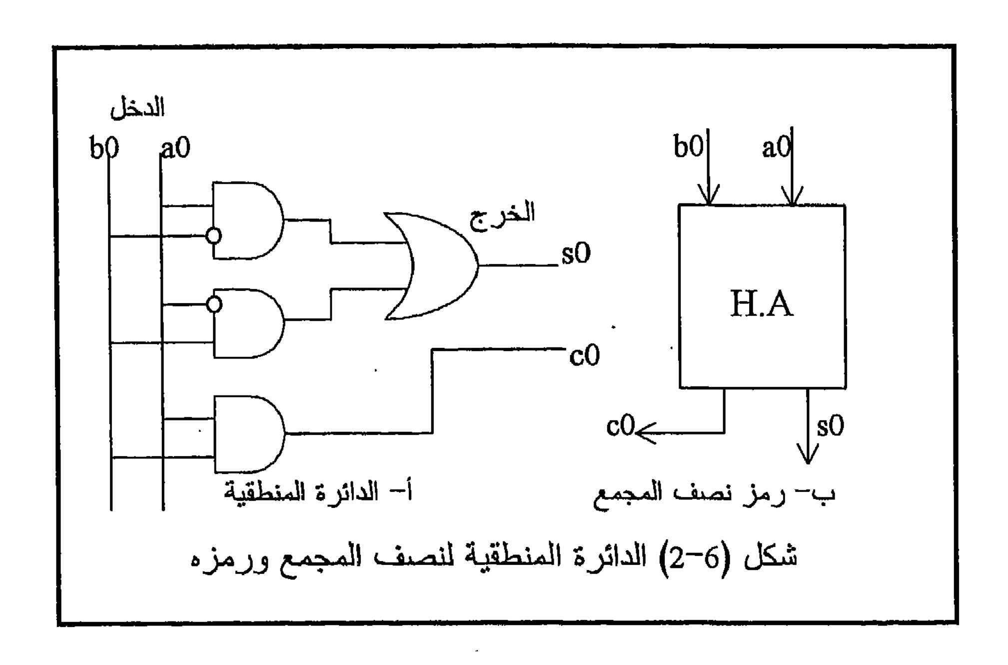
$$s0= a0\,\overline{b0} + \overline{a0}\,b0 \qquad 6-1$$

$$c0= a0b0 \qquad 6-2$$

الخرج		الدخل	
c0	s0	a0	b0
0	0	0	0
0	1	1	0
0	1	0	1
1	0	1	1

شكل (6-1) جدول الحقيقة لنصف المجمع

من المعادلتين 6-1 و 6-2 نستطيع رسم دائرة منطقية لنصف المجمع كما فى الشكل (6-2أ) . انظر أيضا فى نفس الشكل إلى الرمز الذى سنستخدمه لهذه الدائرة .



شكل (6-2) الدائرة المنطقية لنصف المجمع ورمزه

6-2-2 دائرة المجمع الكامل Full adder, FA

دائرة المجمع الكامل تكون قادرة على جمع ثلاثة بتات (cn-1, bn, an) وينتج منها المجموع sn والحمل للمرحلة القادمة cn . جدول الحقيقة لهذه الدائرة موضح فى شكل (6-3) . من جدول الحقيقة نستطيع كتابة معادلات الخرج كما يلى :

$$S_n = a_n \overline{b_n}\, \overline{c_{n-1}} + \overline{a_n}\, b_n\, \overline{c_{n-1}} + \overline{a_n}\, \overline{b_n}\, c_{n-1} + a_n b_n c_{n-1} \qquad 3\text{-}6$$

$$C_n = a_n b_n \overline{c_{n-1}} + a_n \overline{b_n}\, c_{n-1} + \overline{a_n}\, b_n c_{n-1} + a_n b_n c_{n-1} \qquad 4\text{-}6$$

من المعادلتين 6-3 و 6-4 نستطيع استنتاج الدائرة المنطقية للمجمع الكامل كما فى شكل (6-4) . من ذلك نرى أنه لجمع أى رقمين A و B فإننا سنحتاج لنصف مجمع لجمع البت رقم 0 فى كل من الرقمين ثم سنحتاج مجمعا كاملا لجمع كل بت فى الرقم الأول مع ما يناظرها فى الرقم الثانى مع الحمل الناتج من عملية الجمع السابقة . فمثلا لو أن الرقمين A و B كل منهما يتكون من 4 بتات فإننا سنحتاج إلى نصف مجمع وثلاثة مجمعات كاملة لإتمام عملية جمع الرقمين ونفس الكلام يمكن تطبيقه على عملية جمع أى رقمين حيث كل منهما مكون من أى عدد من البتات . شكل (6-5) يبين الدائرة المستخدمة لجمع رقمين كل منهما مكون من أربعة بتات كمثال على ذلك . وترى فى نفس الشكل الرمز العام المستخدم للمجمع .

الخـــــــرج		الـــــدخـــــــل		
cn	sn	an	bn	cn-1
0	0	0	0	0
0	1	1	0	0
0	1	0	1	0
1	0	1	1	0
0	1	0	0	1
1	0	1	0	1
1	0	0	1	1
1	1	1	1	1

شكل (6-3) جدول الحقيقة للمجمع الكامل

3-6 الطرح الثنائى
Binary subtraction

لإجراء عمليات الطرح الثنائى فإنه عادة ما نلجأ إلى تحويل عملية الطرح إلـــى عملية جمع وبعد ذلك يمكن استخدام المجمع الذى سبق شـــرحه لتنفيـذ عمليـة الطرح . لتحويل عملية الطرح إلى عملية جمع ننظر إلى المثال التالى :

مثال 6-2

افترض أن لدينا الرقم A = 1101 فإن المعكـــوس أو المتمـــم الأحـــادى one's complement لهذا الرقم هو $\overline{A} = 0010$ وتم ذلك عن طريق قلب كل 1 إلــى 0 وكل 0 إلى 1 فى الرقم الأصلى . الآن ماذا يحدث لو جمعنا العدد الأصلى زائـد متممه الأحادى زائد واحد كما يلى : ..

$$\begin{array}{r} A = 1101 \\ \overline{A} = 0010 \\ 1+ \\ \hline 1\ 0000 \end{array}$$

الحمل ⟵ 1 0000

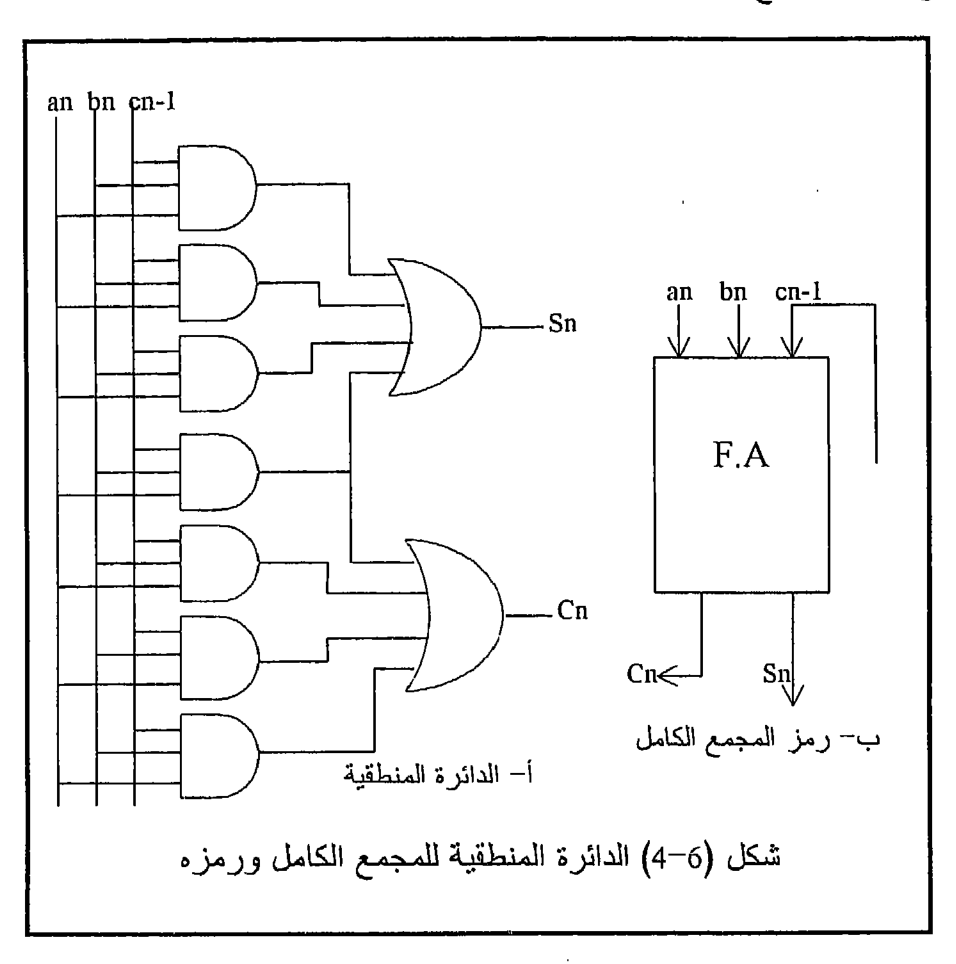
إن النتيجة كما رأينا ستكون دائما صفرا مع حمل واحد ، ولذلك فإنه بإهمال هــذا الحمل يمكننا كتابة العلاقة التالية :

$A + \overline{A} + 1 = 0$

ومنها يمكن كتابة الرقم A على الصورة التالية :

$-A = \overline{A} + 1$ 5–6

وعلى ذلك فإنه من المعادلة (6–5) يمكننا أن نرى أن أى عملية طـــــرح يمكـــن تحويلها إلى عملية جمع عن طريق استبدال المطروح بمتمــــه الثنـــائى (المتمـــم الأحادى + 1) . كمثال على ذلك انظر إلى عمليات الطرح التالية وكيف حولناهـــا إلى عمليات جمع :



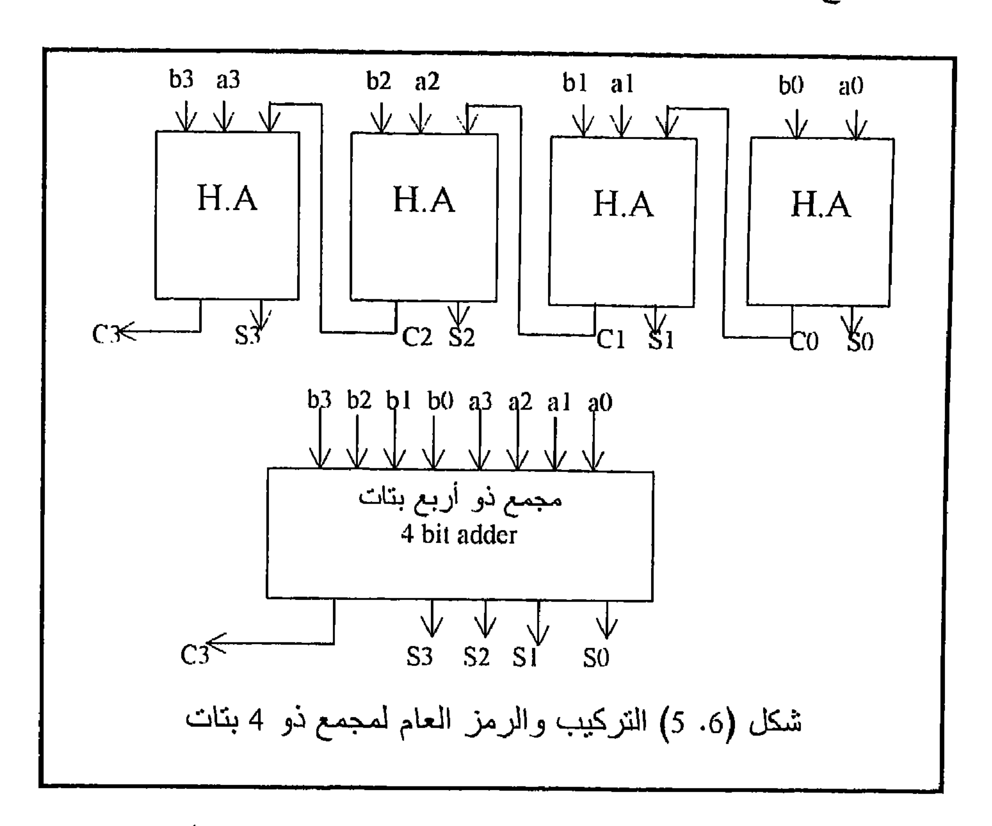
شكل (6–4) الدائرة المنطقية للمجمع الكامل ورمزه

$A - B = A + \overline{B} + 1$ 6–6

$B - C = B + \overline{C} + 1$ 7–6

وبذلك نستطيع القول أنه يمكننا استخدام دائرة المجمع التى سبق شرحها فى تنفيـذ عمليات الطرح أيضا بعد إجراء بعض التعديلات الطفيفة عليـــها . شـــكل (6–6) يبين دائرة مجمع وقد تم عليها هذا التعديل لتقوم بعمليات الجمع أو الطرح عــــن

طريق بعض خطوط التحكم التى يمكن بواسطتها اختيار إما عملية الجمع أو عملية الطرح .

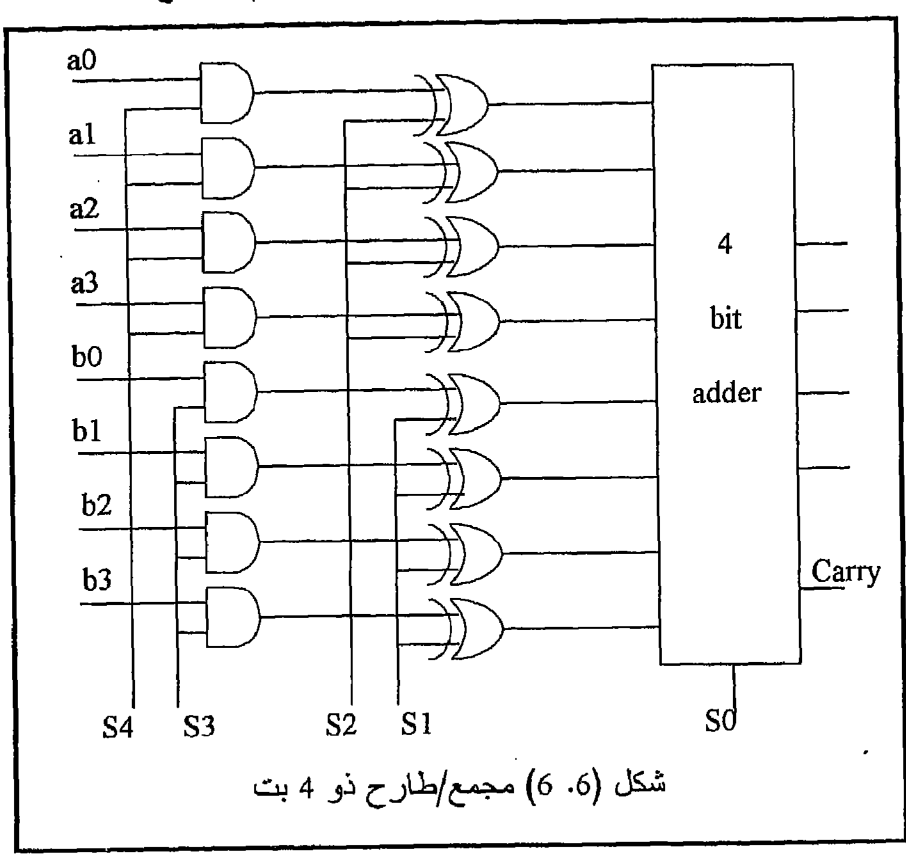


شكل (6. 5) التركيب والرمز العام لمجمع ذو 4 بتات

كما نعلم فإنه من خواص بوابة XOR أن لها دخلان عندما يكون أحدهما يساوى واحدا فإن الخرج يكون معكوس الدخل الآخر ، وعندما يكون أحد هذين الدخلين يساوى صفرا فإن الخرج يكون مثل الدخل الآخر ، أى أن بوابة XOR يمكن استخدام أحد دخليها للتحكم فى جعل الخرج يساوى أما الدخل الآخر أو معكوسه. يمكن أن نرى ذلك بوضوح فى شكل (6-6) بحيث أنه إذا كان خط التحكم S1=1 فإن العدد B سيدخل للمجمع معكوس . أما إذا كان S1=0 فإن العدد B سيدخل إلى المجمع بقيمته الحقيقية . نفس الشيء تم تطبيقه على العدد A باستخدام خط التحكم S2 .

من شكل (6-6) أيضا يمكننا استنتاج وظيفة خطى التحكم S4, S3 بأن كل منهما بمثابة مفتاح (ON/OFF) للعدد الذى يعمل معه . فالخط S3 سيسمح بمرور العدد B أو صفر بدلا منه ، والخط S4 سيسمح بمرور العدد A أو صفر بدلا منه . إنه باستخدام خطوط التحكم S4, S3, S2, S1 يمكن استخدام المجمع الموجود فى شكل (6-6) فى أكثر من وظيفة منها الجمع والطرح . إفترض مثلا أن S4=1,

S1=0, S0=0, S3=1, S2=0, فى هذه الحالة بما أن S4, S3 كل منـــهما يســـاوى واحدا فإن العددين B, A سيعبران من بوابات ال AND بنفس قيمـــهما وبمـــا أن S2, S1 كل منهما يساوى صفرا فإن العددين B, A ســـيعبران مـــن بوابـــات ال XOR بنفس قيمهما الحقيقية ومن ذلك نرى أن المجمع سيكون دخلاه المباشـــران هما العددان B, A وصفر من خط التحكم S0 لذلك سيقوم المجمع بجمعها .



شكل (6 .6) مجمع/طارح ذو 4 بت

افترض الآن الوضع التالى S4=1, S3=1, S2=0, S1=1, S0=1 من شـــكل (6-6) نرى أنه طالما أن S4, S3 كل منهما يساوى واحدا فإن العددان B, A ســـيعبران من بوابات ال AND ، وبما أن S2=0 فإن العدد A سيعبر من بوابـــات ال XOR بقيمته الحقيقية وطالما أن S1=1 فإن العدد B ســـيعبر مـــن بوابـــات ال XOR معكوسا وبذلك نستطيع القول أن المجمع سيقوم بعملية الجمع التالية :

$A + \overline{B} + 1$

حيث الواحد هو قيمة خط التحكم S0 . كما علمنا من قبل فإن $-B=\overline{B}+1$ لذلك فإننا نستطيع القول بأن المجمع فى هذه الحالة يقوم بعملية طرح العدد B من العدد A ، أى (A - B) .
شكل (6-6) به خمسة خطوط تحكم هى S4, S3, S2, S1, S0 كل منها يمكن أن يكون واحدا أو صفرا لذلك فإن هناك 32 (2^5) شفرة أو كود يمكن أن تكون عليها هذه الخطوط ولكل شفرة من هذه الشفرات سيكون هناك خرج معين لدائرة المجمع/الطارح الموضحة فى شكل (6-6) . شكل (6-7) يبين جميع هذه الشفرات والخرج الناتج عن كل منها . إننا هنا لن نقوم بمراجعة جميع الحالات الموجودة فى شكل (6-7) لمعرفة كيف يكون الخرج فى كل حالة ومطابقة ذلك على الدائرة الموجودة فى شكل (6-6) ولكننا سنترك هذه العملية للقارىء اكتفاء بالمثالين اللذين شرحناهما سابقا أحدهما للجمع والآخر للطرح . من الملاحظات المهمة على الخرج أن هناك الكثير من الحالات قد يكون فيها الخرج غير مهم مثل الحالات التى يكون فيها الخرج يساوى 0 أو 1- أو 2- وغير ذلك من الحالات ، ويلاحظ أيضا وجود الكثير من الحالات التى يكون الخرج فيها مكررا مثل الحالة التى يكون فيها الخرج يساوى B قد كررت ثلاث مرات والحالة التى يكون فيها الخرج يساوى 0 كررت أيضا أكثر من مرة ، كيف سنتخلص من هذا التكرار وهذه الحالات التى نعتبرها غير مهمة ؟ إن هذا ما سنراه فى الأجزاء القادمة .

6-4 وحدة الحساب والمنطق
Arithmatic and Logic Unit, ALU

من العمليات التى يقوم بها المعالج دائما بجانب العمليات الحسابية العمليات المنطقية أيضا . لذلك فإننا سنحاول في هذا الجزء وعن طريق إضافة بعض خطوط التحكم أن نجعل الدائرة الموجودة فى شكل (6-6) قادرة على تنفيذ العمليات المنطقية أيضا بجانب العمليات الحسابية (الجمع والطرح) . إن العمليات المنطقية التى سنحاول إضافتها إلى وحدة المجمع/الطارح التى سبق شرحها هى عمليات XOR, OR, AND وهذا كمثال فقط حيث بالطبع يمكن إضافة المزيد .
إن هناك أكثر من طريقة لتطوير دائرة المجمع/الطارح التى سبق شرحها لتستطيع تنفيذ العمليات المنطقية وسنعرض هنا أبسط هذه الطرق . شكل (6-8) يبين وحدة حساب ومنطق تستطيع تنفيذ عمليات الجمع والطرح و OR و XOR و AND . فى هذا الشكل أن الدخلين B, A كل منهما مكون من n من البتات ولكن لتبسيط الرسم تم رسمه كخط واحد فقط . شكل (6-8) مكون من أربعة صناديق كل منها يمثل عملية من عمليات وحدة الحساب والمنطق . الصندوق

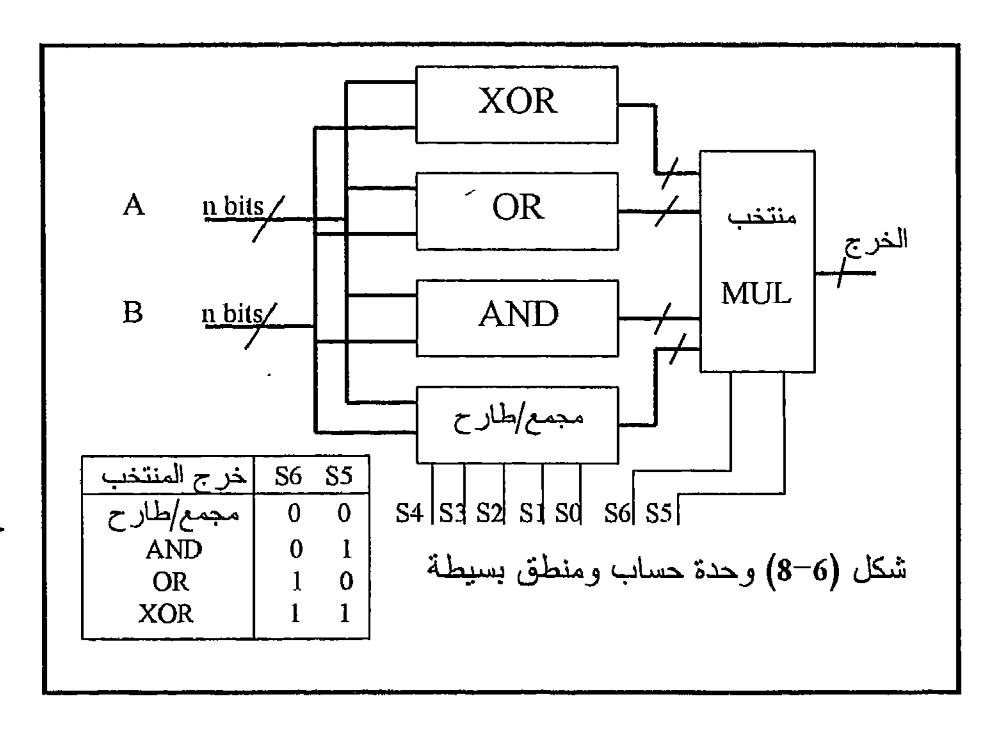
الأول خاص بعمليتى الجمع والطرح ومحتوياته هى الدائرة الموجودة فى شـــكل (6-6) .

s4	s3	s2	s1	s0	الخـــــرج
0	0	0	0	0	0
0	0	0	0	1	1
0	0	0	1	0	-1
0	0	0	1	1	0
0	0	1	0	0	-1
0	0	1	0	1	0
0	0	1	1	0	-2
0	0	1	1	1	-1
0	1	0	0	0	B
0	1	0	0	1	B+1
0	1	0	1	0	-B-1=B
0	1	0	1	1	-B
0	1	1	0	0	B-1
0	1	1	0	1	B
0	1	1	1	0	-B-2
0	1	1	1	1	-B-1=B
1	0	0	0	0	A
1	0	0	0	1	A+1
1	0	0	1	0	A-1
1	0	0	1	1	A
1	0	1	0	0	-A-1=A
1	0	1	0	1	-A
1	0	1	1	0	-A-2
1	0	1	1	1	-A-1=A
1	1	0	0	0	A+B
1	1	0	0	1	A+B+1
1	1	0	1	0	A-B-1
1	1	0	1	1	A-B
1	1	1	0	0	B-A-1
1	1	1	0	1	B-A
1	1	1	1	0	-A-B-2
1	1	1	1	1	-A-B-1

شكل (6-7) جميع الحالات الممكنة لخرج المجمع/الطارح فى شكل (6-6)

الصندوق الثانى خاص بعملية AND ويحتوى عددا n من بوابات AND ثنائية الدخل حيث أحد هذين الدخلين هو بت من بتات الدخل A والدخل الآخر هو البت المناظرة من الدخل B . الصندوق الثالث خاص بعملية OR ومحتوياته هى عدد n من بوابات OR ثنائية المدخل ، وأما الصندوق الرابع فخاص بعملية XOR ومحتوياته هى عدد n من بوابات XOR .
الخرج النهائى لوحدة الحساب والمنطق يتم اختياره من بين خروج الأربعة صناديق السابقة عن طريق منتخب multiplexer له خطى تحكم S6, S5 تتم بهما عملية اختيار أى واحد من الصناديق سيتم توصيل خرجه إلى خرج المنتخب وبالتالى إلى خرج وحدة الحساب والمنطق ، فمثلا إذا كان S5=1, S6=1 فإن خرج صندوق XOR يوصل إلى خرج المنتخب وفى هذه الحالة فإن وحدة الحساب والمنطق ستقوم بتنفيذ عملية XOR على الدخلين B, A . شكل (6–8) يبين أيضا جدول الحقيقة لهذا المنتخب لجميع حالات الخطين S6, S5 . لاحظ أن عملية تخصيص أى شفرة على الخطين S6, S5 ستخرج XOR وأيها ستخرج AND وغير ذلك من العمليات الموضحة فى شكل (6–8) تتوقف على المصمم إذ هو الذى يحدد هذه الشفرات . عندما يكون الخطان S6=0, S5=0 فإن خرج وحدة الحساب والمنطق سيكون هو خرج الصندوق المجمع/الطارح وأما العملية التى ستنفذ فستكون إما عملية جمع أو طرح أو غير ذلك على حسب الشفرة الموجودة على باقى خطوط التحكم S0 إلى S4 .
الدائرة الموجودة فى شكل (6–8) تحتوى على سبعة خطوط تحكم S0 إلى S6 وعلى ذلك فإن هذه الدائرة مفروض أن تكون قادرة على إجراء 128 (2^7) عملية مختلفة بناء على جدول حقيقة يمكن وضعه مماثلا للجدول المبين فى شكل (6–7) . ولكن وكما رأينا فى شكل (6–7) فإن معظم هذه العمليات (128) إما أنها ستكون عمليات غير مهمة أى غير ذات فائدة أو أنها ستكون عمليات مكررة . لذلك فإننا سنقوم بتصفية هذه 128 عملية إلى 13 عملية فقط من العمليات المهمة والغير مكررة وسنهمل باقى العمليات . لاحظ أن اختيار 13 عملية تعتبر من عمل المصمم حيث هو الذى يختار عدد العمليات المهمة وأى العمليات تهمل؟ وأيها يؤخذ فى الإعتبار؟ ولذلك فإن اختيارنا 13 عملية هنا ليس إلا مجرد مثال فقط ولا توجد أى ضرورة لاختيار الرقم 13 بالذات .
بما أننا سنختار 13 عملية فقط وسنهمل الباقى فإنه من البديهى أن يكون هناك 4 خطوط تحكم فقط كافية لتشفير هذه العمليات حيث 2^4 =16 (أكبر من 13) . لذلك فإننا سنحتاج هنا إلى دائرة تكون مهمتها هى تحويل الشفرات الرباعية إلى شفرات سباعية تتناسب مع خطوط التحكم الموجودة فى وحدة الحساب والمنطق المبينة فى شكل (6–8) ، هذه الدائرة سنسميها محول شفرات وعادة ما تكون هذه الدائرة عبارة عن ROM مكونة من 16 بايت ولها أربعة خطوط عناوين تعطى

عليها الشفرة الرباعية فتخرج محتويات البايت المقابلة وهى الشفرة السباعية التــى من المفروض أن تكون مخزنة سلفا .



خرج المنتخب	S6	S5
مجمع/طارح	0	0
AND	0	1
OR	1	0
XOR	1	1

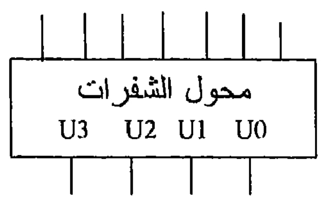
شكل (6-8) وحدة حساب ومنطق بسيطة

شكل (6-9) يبين رسما صندوقيا وجدول الحقيقة لمحول الشفرات . شكل (6-10) يبين الشكل النهائى لوحدة الحساب والمنطق بعد إضافة محول الشفرات إليــــها . لإجراء عملية الجمع مثلا يجب أن نضع الشفرة الرباعية 111 0 علــــى خطـــوط التحكم U3 U2 U1 U0 فيقوم محول الشفرات بإعطاء الشفرة السباعية المنـــاظرة والتى هى الشفرة السباعية لعملية الجمع (011000 0) كما فى جدول الحقيقة فـــى شكل (6-9) .

6-5 مسجل التراكم
Accumulator register

إن إضافة مسجل التراكم Accumulator إلى وحدة الحساب والمنطق هو الخطــوة التالية لتطوير هذه الوحدة وإعدادها لتكون جزءا من أجزاء المعالج . شـــكل (6-11) يوضح هذا التطوير ، من هذا الشكل نلاحظ أن مسجل التراكم يعتبر مخزنــا لتسجيل الناتج من وحدة الحساب والمنطق حيث أنه موصل مباشرة على خرجـــها ولذلك فإننا نتوقع أن ناتج أى عملية حسابية أو منطقية سيذهب مباشرة إلى مسجل

التراكم . نلاحظ أيضا من شكل (6-11) أن هناك نوعا من أنواع التغذية المرتدة حيث قد تم توصيل خرج مسجل التراكم على أحد دخلى وحدة الحساب والمنطق (الدخل A) ولم يبق سوى دخل واحد فقط لوحدة الحساب والمنطق (الدخل B) . نلاحظ أيضا أن مسجل التراكم مثله مثل أى مسجل حيث لن ينتقل دخله إلى خرجه إلا بعد إعطاء نبضة تزامن clock ، لذلك كان لابد من توصيل clock إلى مسجل التراكم .

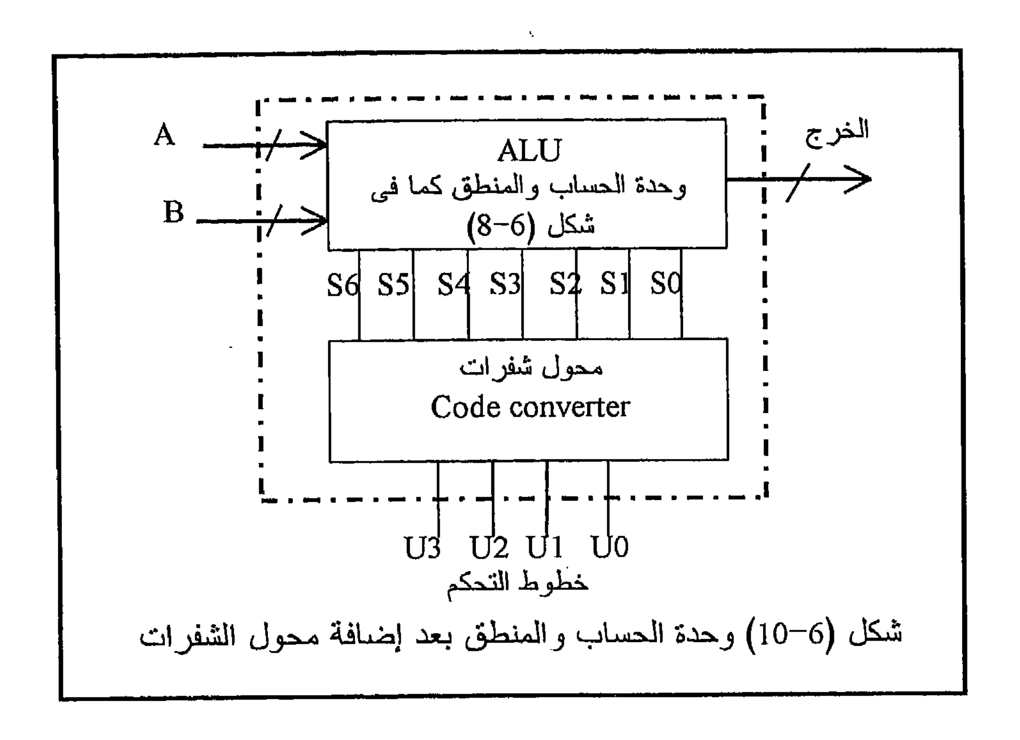


شكل (6-9 أ) رسم صندوقى لمحول الشفرات

U3	U2	U1	U0	العملية	s6	s5	s4	s3	s2	s1	s0
0	0	0	0	A	0	0	1	0	0	0	0
0	0	0	1	1	0	0	0	0	0	0	1
0	0	1	0	A	0	0	1	0	1	0	0
0	0	1	1	B	0	0	0	1	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0
0	1	0	1	A+1	0	0	1	0	0	0	1
0	1	1	0	A-1	0	0	1	0	0	1	0
0	1	1	1	A+B	0	0	1	1	0	0	0
1	0	0	0	A-B	0	0	1	1	0	1	1
1	0	0	1	A∧B	1	0	d	d	d	d	d
1	0	1	0	A∨B	0	1	d	d	d	d	d
1	0	1	1	A+B	1	1	d	d	d	d	d
1	1	0	0	-1	0	0	0	0	1	1	1
1	1	0	1	هذه الحالات الثلاث							
1	1	1	0	غير مستخدمة الآن							
1	1	1	1	وسوف نستخدمها مستقبلا							

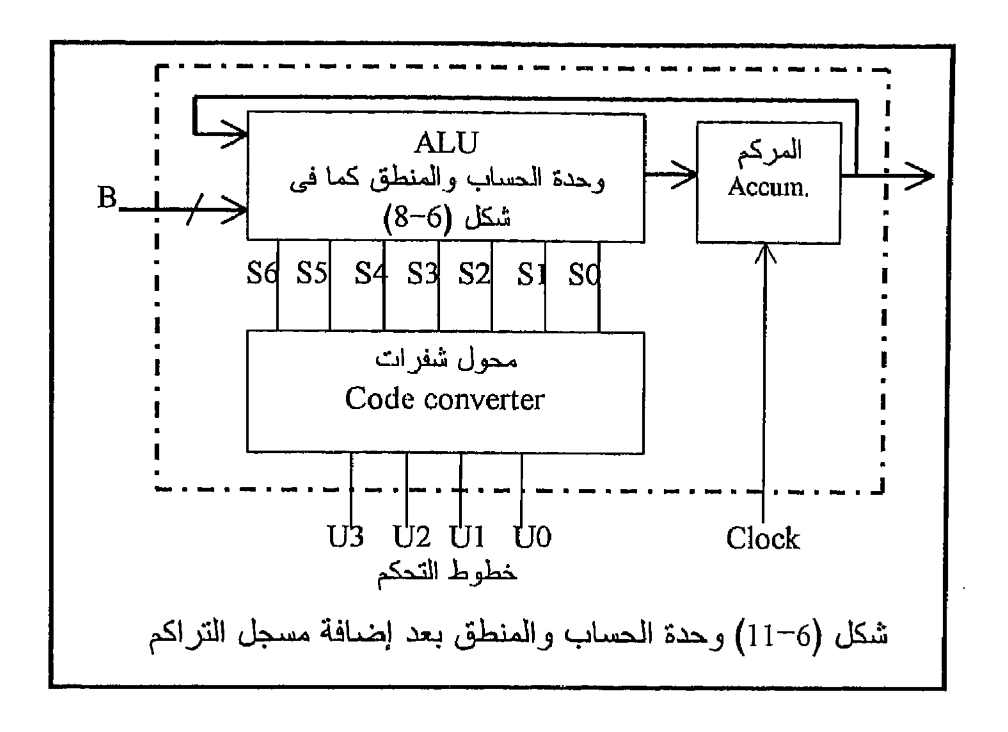
* d تعنى do not care أى لا يهم أن يكون هذا الخط صفر أو واحد .

شكل (6-9ب) جدول الحقيقة لمحول الشفرات



شكل (6-10) وحدة الحساب والمنطق بعد إضافة محول الشفرات

بعد هذا التعديل بإضافة مسجل التراكم إلى وحدة الحساب والمنطق لابد وأن يطرأ تعديل مناظر على ال 13 عملية التى تم تخصيصها لهذه الوحدة للقيام بها والتى سبق أن حددناها سابقا بعد إضافة محول الشفرات فى شكل (6-9) . شكل (6-12) يبين جدولا لهذه العمليات وقد تم إعطاء شفرة حرفية mnomenic أو شفرة أسمبلى لكل عملية بجانب شفرتها الثنائية (الشفرة الرباعية) أو التى سنسميها شفرات الماكينة machine codes . لنأخذ على ذلك المثال التالى : افترض أن خطوط التحكم U3 U2 U1 U0 تساوى 0101 فإنه من شكل (6-9) ستكون العملية (الأمر) التى ستنفذ هى A+1 ، أى أن الدخل A لوحدة الحساب والمنطق الذى هو خرج مسجل التراكم سيزداد بمقدار واحد . بافتراض أن خرج مسجل التراكم كان صفرا فى البداية فإن ال ALU ستجمع واحد زائد صفر وتكون النتيجة واحدا يوضع على خرج ال ALU . مع أول نبضة تزامن فإن هذا الواحد ينتقل إلى خرج مسجل التراكم حيث سيكون موجودا أيضا على الدخل A لل ALU . ومع بقاء نفس الشفرة على الخطوط U0 إلى U3 فإن هذا الواحد سيصبح اثنين على خرج ال ALU . بإعطاء نبضة التزامن الثانية ستنتقل الاثنين إلى خرج مسجل التراكم والدخل A لل ALU وهكذا تستمر العملية . أى أن هذا الأمر يزيد واحدا على محتويات مسجل التراكم مع كل نبضة تزامن كما لو كان عدادا ولذلك فقد تم اختصاره إلى INC أى زد بمقدار واحد Increment .



شكل (6-11) وحدة الحساب والمنطق بعد إضافة مسجل التراكم

كما نلاحظ من شكل (6-11) فإن خرج مسجل التراكم متصل دائما بالدخل A لوحدة الحساب والمنطق لذلك فإنه دائما يطلق عليه إسم المسجل A ، وبما أن النتائج دائما تتراكم فيه كما رأينا فى المثال السابق فقد أطلق عليه أيضا اسم مسجل التراكم أو المركم Accumulator . لنأخذ مثالا آخر أكثر تعقيدا لكى نفهم كيفية عمل الدائرة الموجودة فى شكل (6-11) . افترض أن المطلوب هو ضرب العددين 3×6 . كما نرى من شكل (6-12) فإنه ليس هناك أى أمر يقوم بعملية الضرب ، لذلك فإننا سننفذ الضرب عن طريق الجمع المتكرر ، أى أننا سنجمع العدد 6 مع نفسه ثلاث مرات . لذلك فإننا سنضع العدد 6 على الدخل B أولا ثم سنضع الشفرة 0011 على خطوط التحكم U0 إلى U3 وهذه الشفرة من شأنها أن تنقل الموجود على الدخل B إلى خرج مسجل التراكم مع أول نبضة تزامن ، أى أنه بعد أول نبضة تزامن سيكون العدد 6 موجودا على خرج مسجل التراكم وبالتالي أيضا على الدخل الآخر (الدخل A) لوحدة الحساب والمنطق . الآن سنضع الشفرة 0111 على خطوط التحكم U0 إلى U3 وهذه الشفرة كما فى شكل (6-12) ستقوم بجمع محتويات الدخل B مع محتويات الدخل A لـ ALU . وعلى ذلك فإنه بإعطاء نبضة تزامن ثانية ستكون محتويات مسجل التراكم تساوى12 وهى حاصل جمع 6 الموجودة على الدخل B و 6 الموجودة على خرج مسجل التراكم بعد النبضة الأولى . لاحظ أن الدخل A لـ ALU بعد

النبضة الثانية سيصبح 12 أيضا . لو احتفظنا بنفس الشفرة 0111 علـــى خطـــوط التحكم U0 إلى U3 وأعطينا نبضة تزامن أخرى فـــإن خـــرج مســـجل الـــتراكم سيصبح 18 وهى حاصل جمع 6 الموجودة على الدخل B و 12 الموجودة علـــى خرج مسجل التراكم بعد النبضة الثانية . بذلك نكون قد أنـــهينا عمليـــة ضـــرب العددين 3×6 بعد ثلاث نبضات تزامن وبعد أن وضعنا الشـــفرات التاليـــة علـــى خطوط التحكم U0 إلى U3 :

	U3	U2	U1	U0
أول نبضة تزامن	0	0	1	1
ثانى نبضة تزامن	0	1	1	1
ثالث نبضة تزامن	0	1	1	1

من المثال السابق يمكننا أن نقول أن أى عملية مركبـــة يمكـــن تبســـيطها إلـــى مجموعة من الشفرات التى توضع على خطوط التحكم U0 إلى U3 واحدة بعـــد الأخرى بحيث تنفذ العملية المناظرة لكل شفرة مع نبضة تزامن . الآن ألا يمكننـــا أن نسمى كل شفرة من هذه الشفرات بالأمر Instruction ، ومجموعة الشـــفرات أو الأوامر المطلوبة لتنفيذ أى عملية مركبة ألا نسميها برنـــــامج program ، ألا يمكن أن نسمى الدائرة التى لها قائمة الأوامـــر الموجـــودة فـــى شـــكل (6−12) بالمعالج . نعم إن الدائرة التى حصلنا عليها فى شـــكل (6−11) تعتـــبر صـــورة مبسطة جدا لمعالج افتراضى محدود الإمكانيات إذا ما قورن بأى معالج حقيقى . إن المعالج الإفتراضى hypothetical الذى وصلنا إليه حتى الآن يستطيع فقـــط إجراء العمليات الحسابية والمنطقية ، السؤال الآن هل نستطيع تطوير الوحـــدة السابقة لكى تستطيع التعامل مع ذاكرة ووحدات إدخال أو إخراج كما فى المعالج العادى ؟ إن هذا ما سنراه فى الجزء التالى :

6−6 إضافة ذاكرة للمعالج الإفتراضى

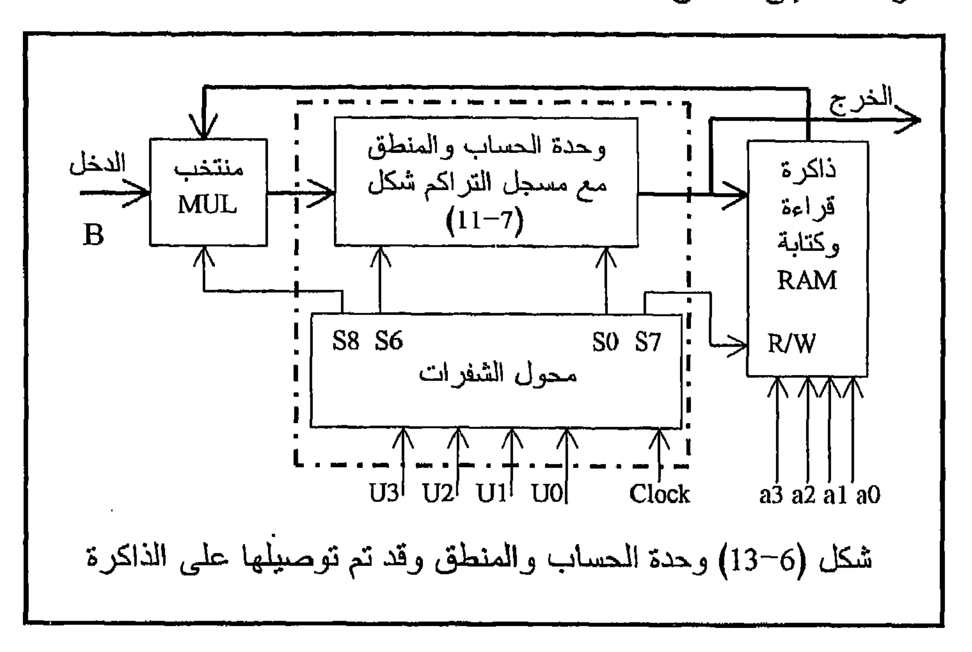
بالنظر إلى الدائرة الموجودة فى شكل (6−13) نجد أن هناك منتخبا تمت إضافته فى الدخل وهذا المنتخب له دخلان وخط تحكم واحد S8 تمت إضافته على خرج محول الشفرات لهذا الغرض . أحد دخلى هذا المنتخب هـــو خـــرج الذاكـــرة والدخل الآخر هو الدخل B حيث يتم اختيار أحدهما ليعبر إلى خرج المنتخـــب ومنه كدخل إلى وحدة الحساب والمنطق ثم إلى مسجل التراكم عن طريق خـــط التحكم S8 . وعلى ذلك فعندما يكون S8=0 فإن خرج الذاكرة يكون موصلا إلى وحدة الحساب والمنطق ، وأما إذا كان S8=1 فإن الدخل B يوصل إلـــى وحـــدة الحساب والمنطق .

U3 U2 U1 U0	اختصار العملية	العمليـــــــة	الحمل Carry
0 0 0 0	NOP	لاتعمل شيء	no
0 0 0 1	SP1	A = +1	no
0 0 1 0	CMA	اعكس A	no
0 0 1 1	LDA	حمل A بالدخل B	no
0 1 0 0	CLA	A = 0	no
0 1 0 1	INC	$A+1 \rightarrow A$	yes
0 1 1 0	DEC	$A-1 \rightarrow A$	yes
0 1 1 1	ADD	$A+B \rightarrow A$	yes
1 0 0 0	SUB	$A-B \rightarrow B$	yes
1 0 0 1	AND	$A \wedge B \rightarrow A$	no
1 0 1 0	OR	$A \vee B \rightarrow A$	no
1 0 1 1	XOR	$A \oplus B \rightarrow A$	no
1 1 0 0	SM1	$-1 \rightarrow A$	no
1 1 0 1			
1 1 1 0			
1 1 1 1			
* A مقصود بها مسجل التراكم و B يقصد بها الدخـــل B لوحـــدة الحساب والمنطق			

شكل (6-12) قائمة بالعمليات التى تنفذها وحدة الحساب والمنطق التى معها مركم كما فى شكل (6-11)

المشكلة الآن أننا نريد توصيل أحد هذين الدخلين إلـــى مســجل الــتراكم فــأى الشفرات نضعها على خطوط التحكم U0 إلى U3 ؟ ليس هناك الا شفرة واحـــدة فقط وهى الشفرة 0011 التى تضع دخل وحدة الحساب والمنطـــق فـــى مســجل التراكم وهى شفرة الأمر LDA وكما رأينا فإن دخل وحدة الحساب والمنطق إمــا أن يكون من الذاكرة أو من B . لذلك فإنه لتمييز كل من الدخلين من الآخر فإننــا سنخصص الشفرة 0011 التى هى الأمر LDA لتحميل المركم بمحتويات دخـــل وحدة الحساب والمنطق إذا كان هذا الدخل قادما من الذاكرة أى S8=0 . ولذلــك فإنه فى هذه الحالة(S8=0) لابد من تحديد العنوان الذى سيتم التعامل معه علـــى خطوط عنوان الذاكرة a0 إلى a3 . أما إذا كان الدخل قادما من B فإننا سنستخدم له شفرة من الشفرات الغير مستخدمة والمتبقية من ال 16 شفرة الموجـــودة فـــى قائمة الأوامر ولتكن الشفرة 1101 والتى سنعطيها الاختصار INP أى الدخل مـــن

خارج المعالج مثل بوابة إدخال مثلا . لاحظ أنه مع الأمر INP فإن خط التحكم S8 يكون واحدا فى هذه الحالة وستكون الشفرة المقابلة على خرج محول الشفرات S0 إلى S8 هى 00001000 .



شكل (6-13) وحدة الحساب والمنطق وقد تم توصيلها على الذاكرة

بذلك نكون قد رأينا كيفية اختيار دخل وحدة الحساب والمنطق بين الذاكرة أو الدخل B القادم من خارج المعالج ، ماذا عن خرج وحدة الحساب والمنطق وكيف يمكن توصيله إلى الذاكرة ؟

إن خط التحكم S7 الذى أضيف على خرج محول الشفرات كما فى شكل (6-13) مهمته هى التحكم فى كتابة خرج مسجل التراكم فى الذاكرة إذا كان S7=1 أو إخراجه إلى خارج المعالج عندما يكون S7=0 . لاحظ أن كلا من هاتين العمليتين ليس لها أمر يقابلها فى قائمة الأوامر التى درسناها حتى الآن فى شكل (6-12) ولذلك فإننا سنستخدم الشفرتين المتبقيتين من ال 16 شفرة فى نفس الشكل لهذا الغرض . أى أن الشفرة 1110 والتى سنختصرها STA أى خزن فى الذاكرة Store accumulator ستجعل الخط S7=1 الذى سيجعل الذاكرة فى حالة استقبال للمعلومات أى فى حالة كتابة فيها لأن الخط R/W=1 . لاحظ أن هذا الأمر سيتطلب أن نضع العنوان الذى سيتم التعامل معه داخل الذاكرة على مسار العناوين (a0 إلى a3) .

U3 U2 U1 U0	العنوان a3 a2 a1 a0	الأمر	العملية	الحمل
0 0 0 0	x x x x	NOP	لاتعمل شيء	no
0 0 0 1	x x x x	SP1	A = +1	no
0 0 1 0	x x x x	CMA	اعكس A	no
0 0 1 1	a a a a	LDA	A ⟵ الذاكرة	no
0 1 0 0	x x x x	CLA	A = 0	no
0 1 0 1	x x x x	INC	A+1 ⟶ A	yes
0 1 1 0	x x x x	DEC	A-1 ⟶ A	yes
0 1 1 1	a a a a	ADD	A ← A+عنوان	yes
1 0 0 0	a a a a	SUB	A ← A -عنوان	yes
1 0 0 1	a a a a	AND	A ⟵ A∧عنوان	no
1 0 1 0	a a a a	OR	A ← A∨عنوان	no
1 0 1 1	a a a a	XOR	A ⟵ A⊕عنوان	no
1 1 0 0	x x x x	SM1	-1 ⟶ A	no
1 1 0 1	x x x x	INP	B ⟶ A	no
1 1 1 0	a a a a	STA	الذاكرة ← A	no
1 1 1 1		OUT	الخرج ⟵ A	no

شكل (6-14) قائمة الأوامر بعد تعديلها لتلائم عملية الإتصال بالذاكرة

أما الشفرة 1111 المتبقية فسوف نخصصها لإخراج محتويات المركم خارج المعالج ولذلك سنرمز لها بالرمز OUT أى إخراج output حيث سيكون الخط S7=0 فى هذه الحالة . شكل (6-14) يبين قائمة الأوامر بعد تعديلها لتناسب عملية التوصيل للذاكرة وإضافة الأوامر الجديدة إليها .

إننا بعد دراسة هذا الفصل يجب أن نقف وقفة تفكير فى الفرق بين هذا المعالج الإفتراضى وأى واحد من المعالج الحقيقى الذى درسناه فى الفصول السابقة . أسئلة كثيرة يمكن أن ترد هنا بعد دراسة الأشكال المختلفة والمتعددة التى رأيناها فى هذا الفصل . كيف سيكون شكل المعالج لو أننا رسمناه على أساس 8 بتات أو 16 أو حتى 32 بت ؟ وماذا لو جعلنا الشفرة الداخلة إلى محول الشفرات شفرة من 16 بت بدلا من الشفرة الرباعية ؟ وكيف سيكون شكل هذه الدائرة إذا أضفنا بعض العمليات الأخرى مثل عمليات دوران محتويات المسجل A إلى اليمين أو إلى اليسار وعمليات الإزاحة من اليمين أو اليسار وعمليات الضرب والقسمة ؟ كيف سيكون شكل هذه الدائرة لو أضفنا لهذه الوحدة بعض المسجلات عامة الأغراض التى تستخدم لأغراض التخزين وتبادل المعلومات مع المسجل A ؟

كيف سيكون شكل الدائرة لو أن الوحدة تتعامل مع ذاكرة لها 16 أو 32 خطا من خطوط العناوين بدلا من 4 كما افترضنا ؟ كل هذه أسئلة توضح مدى بساطة فكرة عمل المعالج ولكنها فى نفس الوقت توضح مدى تعقيد تركيبه ومكوناته .

6-7 تمارين

1. دائرة نصف المجمع الموجودة فى شكل (6-2) يمكن بناؤها باستخدام بوابات XOR ، وضح ذلك ؟
2. دائرة المجمع الكامل الموجودة فى شكل (6-4) يمكن بناؤها باستخدام نصفى مجمع ، وضح ذلك ؟
3. هناك بعض الشرائح التى تعمل كمجمع ، اذكر واحدة من هذه الشرائح واشرح طريقة عملها واختبرها معمليا؟ استعن بكتاب data book دليل الشرائح.
4. يلعب المتمم الثنائى دورا مهما فى تحويل عملية الطرح إلى عملية جمع ، وضح ذلك بالشرح ؟
5. ما هو دور وحدة الحساب والمنطق فى المعالج ؟
6. هناك بعض الشرائح التى تعمل كوحدة حساب ومنطق ، اذكر واحدة من هذه الشرائح ، واشرح طريقة عملها واختبرها معمليا ؟
7. شكل (6-9) يبين محول شفرات رباعية إلى سباعية ، ماذا يحدث لو استخدمنا محول شفرات خماسية إلى سباعية بدلا منه ؟ اكتب قائمة العمليات الجديدة بعد العمليات الاضافية ؟
8. هل مسجل التراكم الموجود فى شكل (6-11) يقوم بنفس دور مسجل التراكم فى أى معالج من حيث أنه يكون طرفا فى أى عملية حسابية أو منطقية والنتيجة تخزن فيه ؟
9. ما مقدار الذاكرة التى يستطيع المعالج الافتراضى الموجود فى شكل (6-13) أن يتعامل معها ؟ وكيف نزيد هذه الكمية ؟
10. لو أردنا إضافة مسجل أوامر للوحدة الموجودة فى شكل (6-13) فأين يكون ، وضح ذلك ؟

الفصل السابع

أساسيات مواجهة المعالج

Fundamentals of Microprocessor Interfacing

7-1 مقدمة

مهمتان أساسيتان تستطيع عملهما باستخدام المعالج : المهمة الأولى هى أنـــك، تستطيع برمجته لحل أى مشكلة إذا كان كل ما تطلبه هذه المشكلة هو البرمجـــة فقط (software) . المهمة الثانية التى تستطيع عملها باستخدام المعالج هـــى أنــك تستطيع مواجهته مع بعض الدوائر الخارجية للوصول إلى الكثير من الأغـــراض والتى سنذكر منها اثنين فقط على سبيل المثال لا الحصر:

1. تستطيع مواجهة المعالج مع بعض الدوائر الخارجية مثـــل دوائـــر الذاكـــرة وبوابات الإدخال وبوابات الإخراج لعمل حاسب شخصى للأغراض العامة مثـــل الحاسبات التى نراها كثيرا فى حياتنا اليومية والتى ما منها حاسب إلا قائم أساســا على معالج معين .
2. إنك تستطيع مواجهة المعالج مع بعض الشرائح الخارجية مثل شرائح الذاكـــرة وبوابات الإدخال والإخراج للحصول على نظام تحكم فى عملية صناعية معينة .

لكى يتم التحكم فى أى عملية صناعية باستخدام المعالج (كالتحكم فـــى ســـرعة محرك مثلا) ليس من الضرورى أن نضع بجوار هذه العملية كومبيوتر متكـــامل باهظ الثمن لإتمام عملية التحكم ولكننا نستطيع بناء دائرة مبسطة علـــى كـــارت واحد مكونة من المعالج وعدد قليل من الشرائح الموصلة عليه لاســـتيفاء هـــذا الغرض وبأقل التكاليف الممكنة .

إننا فى هذا الفصل وقبل أن ندخل فى تفاصيل عملية مواجهـــة أى معـــالج مـــع الدوائر الخارجية سنحاول أولا التعرف على بعض الأساسيات الضرورية والتـــى يجب أن نأخذها فى الاعتبار قبل البدء فى عملية المواجهة مع أى معالج .

7-2 فصل خطوط المعالج
Buffering of microprocessor lines

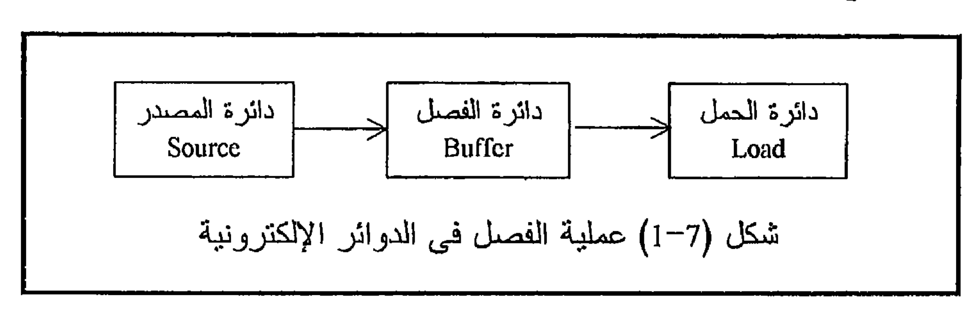
7-2-1 ماذا نعنى بكلمة فصل ؟

إن الفصل يكون عادة بين شيئين ينتج من اتصالهما المباشر بعـــض المشـــاكل ، لذلك فإننا نضع بينهما وسيطا يكون حلقة الوصل بين هذين الشـــيئين . وأقـــرب مثال على ذلك ما نسمعه فى عصرنا الحالى الملىء بالحروب عن وجود منطقـــة فاصلة بين القوتين المتحاربتين تتمركز فيها قوات محايدة تكون حلقة الوصل بيـــن القوتين . هذا الموقف ينشأ فى الكثير من الدوائر الإلكترونية عند تحميل إحداهـــا على الأخرى . ماذا يحدث لو أن الدائرة المصدر كانت غير قـــادرة علـــى إدارة driving أو تشغيل الدائرة الحمل بسبب أن الدائرة الحمل تحتاج إلى الكثـــير مـــن التيار الذى لا تستطيع الدائرة المصدر توفيره ؟ الذى يحدث هو أن جـــهد خـــرج

الدائرة المصدر يضمحل أو يتلاشى وبذلك تصبح الدائرة غير قادرة على إدارة الحمل . فى هذه الحالة يكون الحل هو استخدام فاصل Buffer بين الدائرتين ويكون هذا الفاصل عبارة عن دائرة إلكترونية (وليس قوات أمم متحدة) تستطيع الدائرة المصدر إدارتها وتستطيع هى إدارة الدائرة الحمل ، ونؤكد هنا على أن الدائرة الفاصلة تكون دائرة يستطيع المصدر إدارتها وإلا فليس هناك أى معنى لاستخدامها كفاصل لأنها فى هذه الحالة ستحتاج إلى فاصل . شكل (7-1) يبين عملية الفصل بين دائرتين باستخدام دائرة فصل .

7-2-2 متى نحتاج لفصل buffering خطوط المعالج ؟

من المعروف وكما سنرى فى الفصول القادمة أن جميع خطوط المعالج الخارجة منه توصل على الكثير من الدوائر أو الشرائح الإلكترونية على التوازى ، فمثلا خطوط العناوين توصل على العديد من شرائح الذاكرة سواء RAM أو ROM والعديد من بوابات الإدخال وبوابات الإخراج . جميع هذه الشرائح تعتبر أحمالا بالنسبة للمعالج عليه الوفاء باحتياجاتها من التيار ، فعندما يكون خط العنوان الخارج من المعالج High واحد فإن هذه الشرائح ستسحب تيارا معينا من المعالج لابد وأن يستطيع توفيره ، وعندما يكون خط العنوان الخارج من المعالج low أى صفر فإن هذه الشرائح ستصرف تيارات معينة لابد وأن يكون المعالج قادرا على صرفها أو بلعها .



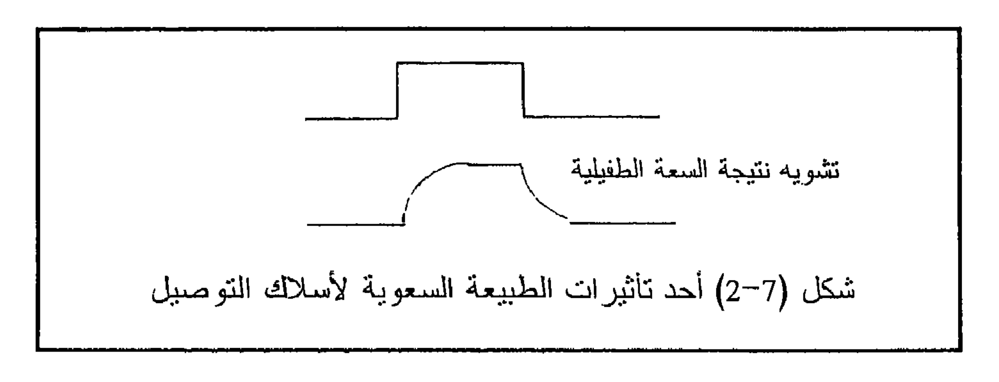
شكل (7-1) عملية الفصل فى الدوائر الإلكترونية

بتجميع هذه التيارات الداخلة والخارجة على الخطوط المتناظرة لجميع الشرائح الموصلة على المعالج يمكننا أن نعرف كم من التيار يجب على شريحة المعالج أن توفرها للشرائح الخارجية فى حالة high وكم من التيار يجب أن تستقبلها أو تصرفها sinking فى حالة low بعد حساب مجموع التيارات المطلوب توفيرها أو صرفها من المعالج يقوم المصمم بالنظر فى الكتالوج الخاص بشريحة المعالج لمعرفة هل سيستطيع الوفاء بهذه الأغراض أم لا ؟ بناء على ذلك سيأخذ المصمم قراره بالحاجة إلى فاصل أم لا بناء على المواقف التالية :

1. إذا وجد المصمم أن احتياجات الأحمال من التيار ليست أقل مما يستطيع

المعالج توفيره وبكمية كافية كعامل أمان فإنه فى هذه الحالة لابد من اللجوء إلى استخدام فاصل buffer ، ويجب على المصمم ألا يضع نفسه فى المنطقة الحرجة كأن يهمل استخدام الفاصل إذا وجد أن التيارات التى تحتاجها الأحمال أقل أو تكاد تساوى ما يستطيع المعالج الوفاء به ، بل يجب عليه أن يعطى نفسه عامل أمان كافي لأن ذلك بالطبع سيؤثر على تشغيل الدائرة فيما بعد .

2. إذا كانت المسافة بين الحمل والمعالج طويلة بحيث سيحتاج الوضع لاستخدام أسلاك توصيل cables طويلة لإتمام عملية التوصيل ، فإنه فى هذه الحالة لابد من استخدام دائرة فصل عند الخروج من المعالج وقبل السلك . إن ذلك ناشئ من الطبيعة السعوية لسلك التوصيل ، فمثل هذه الأسلاك تكون لها سعة capacitance وهذه السعة قد تؤثر على شكل الموجة الخارجة من المعالج وتشوهها وبالذات فى مثل هذه التطبيقات التى تكون الموجات المربعة هى المستخدمة عادة وحيث تكون ترددات هذه الموجات عالية بحيث تتأثر بمثل هذه السعة الناشئة عن السلك . شكل (2-7) يبين موجة مربعة وقد حصل لها تشويه بسبب السعة الطفيلية parasitic capacitance للسلك . لذلك فإنه عامة نستطيع القول أنه إذا كان الحمل موجودا على كرت غير الكرت الذى عليه المعالج فإنه فى هذه الحالة يستحسن استخدام فواصل buffers على جميع المسارات بعد المعالج مباشرة .



شكل (2-7) أحد تأثيرات الطبيعة السعوية لأسلاك التوصيل

3. هناك بعض المعالجات التى تستخدم فكرة المزج الزمنى time multiplexing بين مساراتها مثل المعالج 8085 الذى يستخدم الثمانية خطوط الأولى من مسار العناوين كمسار للبيانات أيضا بحيث أن هذه الخطوط تحمل إشارة عناوين لمدة معينة من الزمن ثم بعد ذلك تحمل إشارة بيانات كما سنرى ذلك بالتفصيل فى الفصل القادم . لمثل هذه المعالجات لابد من إجراء عملية عزل لإشارة البيانات وحدها لتكون على مسار خاص و لإشارة العناوين وحدها لتكون على مسار آخر وذلك قبل توصيل الأحمال على هذه المسارات . إذا كان ذلك سيتم فإنه عادة

يستخدم شرائح تقوم بعملية عزل أو فصل للإشارات كل على مسار خاص وفى نفس الوقت تكون هذه الشرائح فواصل buffers تفى بأغراض الأحمال من التيارات .

الآن إذا تم أخذ القرار بأنه لابد من استخدام الفواصل فهناك بعض الملاحظات التى يجب أن تؤخذ فى الاعتبار عند اختيار الشريحة التى ستستخدم كفاصل لأن هناك الكثير من الشرائح التى تستطيع القيام بهذه المهمة :

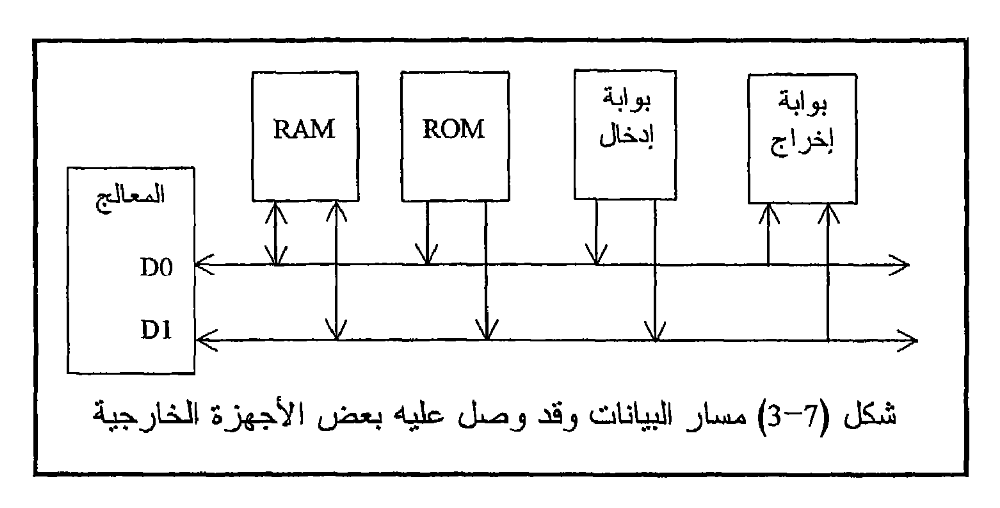
1. بالطبع كما ذكرنا فإن الفاصل buffer الذى ستستخدمه لابد وأن يكون قادرا على الوفاء بالتزامات التيار المطلوبة للأحمال وإلا فلا فائدة من استخدامه .
2. لابد أن يكون المعالج يستطيع إدارة جميع الفواصل المركبة على خطوطه وإلا أيضا فلا فائدة من استخدامها .
3. يجب ألا تؤثر الفواصل على طبيعة الإشارة فهناك مثلا فواصل من طبيعتها أنها تعكس الإشارة (تجعل الواحد صفرا والصفر واحدا) لذلك يجب أن يؤخذ ذلك فى الاعتبار إذا كانت مثل هذه الفواصل سوف تستخدم .
4. يجب أن يناسب الفاصل buffer طبيعة الإشارة التى ستمر من خلاله ، فهناك مثلا مسارات أحادية الاتجاه بمعنى أن الإشارة عليها تمر فى اتجاه واحد فقط مثل مسار العناوين الذى تكون الإشارة عليه دائما من المعالج إلى الأجهزة الخارجية لذلك يجب على الفاصل أن يسمح بذلك ، كذلك فإن من طبيعة مسار البيانات أن الإشارة عليه تمر فى كلا الاتجاهين من وإلى المعالج لذلك أيضا يجب على الفاصل المستخدم أن يأخذ ذلك فى الاعتبار .

7-3 البوابات ثلاثية المنطق
Tristate logic gates

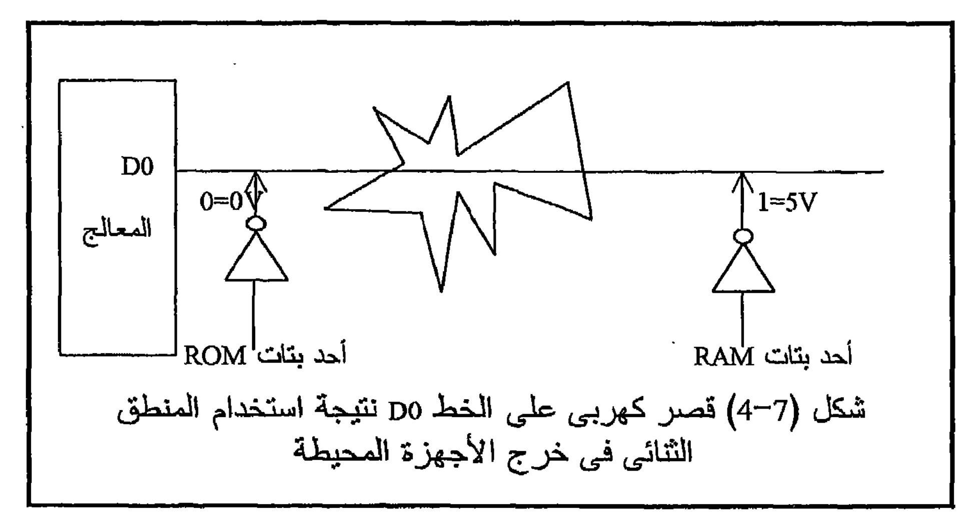
إن أى مسار من مسارات المعالج يكون عبارة عن مجموعة من الخطوط المتوازية التى أصلها ، أى تخرج من المعالج ، وموصل عليها على التوازى الكثير من الأجهزة الخارجية مثل شرائح RAM و ROM وبوابات الإدخال والإخراج . من هذه المسارات مسار البيانات الذى يتكون من ثمانية خطوط فى بعض شرائح المعالجات التى ندرسها فى هذا الكتاب . شكل (7-3) يبين هذا المسار خارجا من المعالج وقد وصل عليه الكثير من الأجهزة المحيطة .

فى حالة المنطق الثنائى سنجد أن جميع هذه الأجهزة لابد وأن يكون خرجها إما صفرا أو واحدا . لذلك فإنه من الممكن أن يكون الخط D0 مثلا من مسار البيانات عليه صفرا من خرج ال RAM وفى نفس الوقت يكون عليه واحدا من خرج ال ROM ، وكما نعلم فإن الواحد المنطقى فى نظام ال TTL يعنى جهدا أكثر من 2.4 فولت والصفر المنطقى يعنى جهدا أقل من 0.4 فولت ووجود هذين

الجهدين على نفس الخط وفى نفس الوقت يعنى قصر كهربى short circuit مما سينتج عنه تخريب لمرحلة الخرج فى أحد الجهازين إن لم يكن كليهما . شكل (7-4) يبين خط البيانات D0 وقد حدث عليه هذا القصر short circuit نتيجة توصيل الجهازين اللذين خرجهما عبارة عن مرحلة ثنائية المنطق التى تكون إما صفر أو واحد .



شكل (7-3) مسار البيانات وقد وصل عليه بعض الأجهزة الخارجية



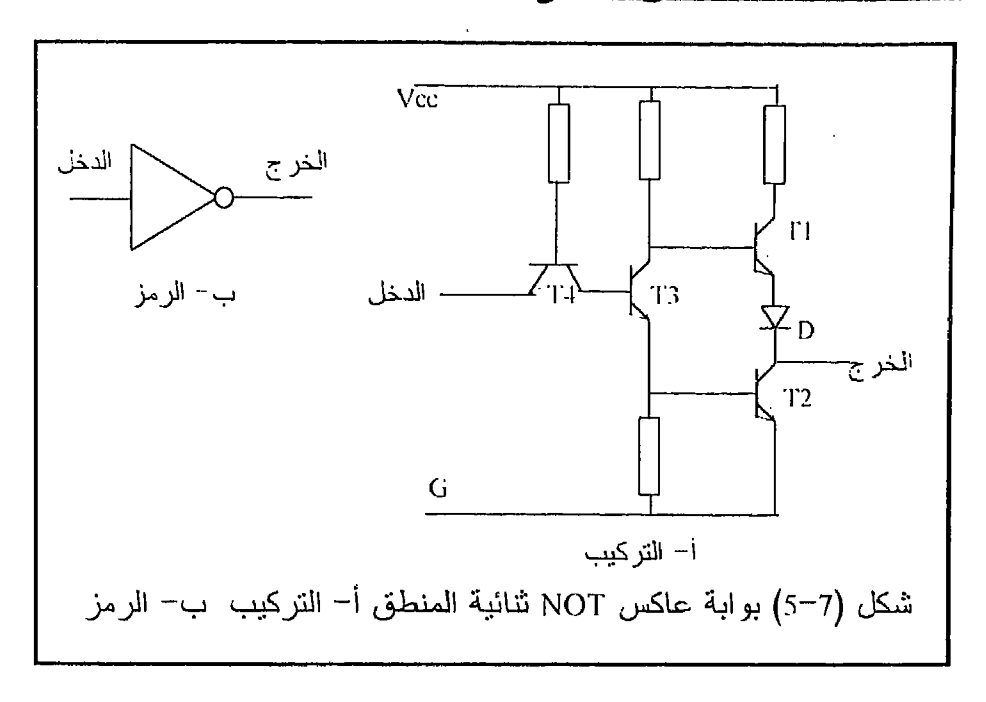
شكل (7-4) قصر كهربى على الخط D0 نتيجة استخدام المنطق الثنائى فى خرج الأجهزة المحيطة

الآن ما هو الحل لهذه المشكلة ؟

إن هذه المشكلة يمكن أن تتلاشى إذا استخدمنا فى مرحلة خرج هذه الأجهزة بوابات ثلاثية المنطق بدلا من البوابات ثنائية المنطق التى نتجت عنها هذه المشكلة . لكى نفهم تركيب البوابات ثلاثية المنطق سنأخذ نظرة سريعة على

تركيب البوابات ثنائية المنطق ولكن قبل ذلك يجب أن نعى جيدا أن هذه المشكلة مصاحبة فقط للأجهزة التى تقوم بإدخال معلومات إلى المعالج من خلال مسار البيانات مثل بوابات الإدخال وال RAM وال ROM ، وأما الأجهزة التى تستقبل معلومات من المعالج مثل بوابات الإخراج فلا تعانى من هذه المشكلة .

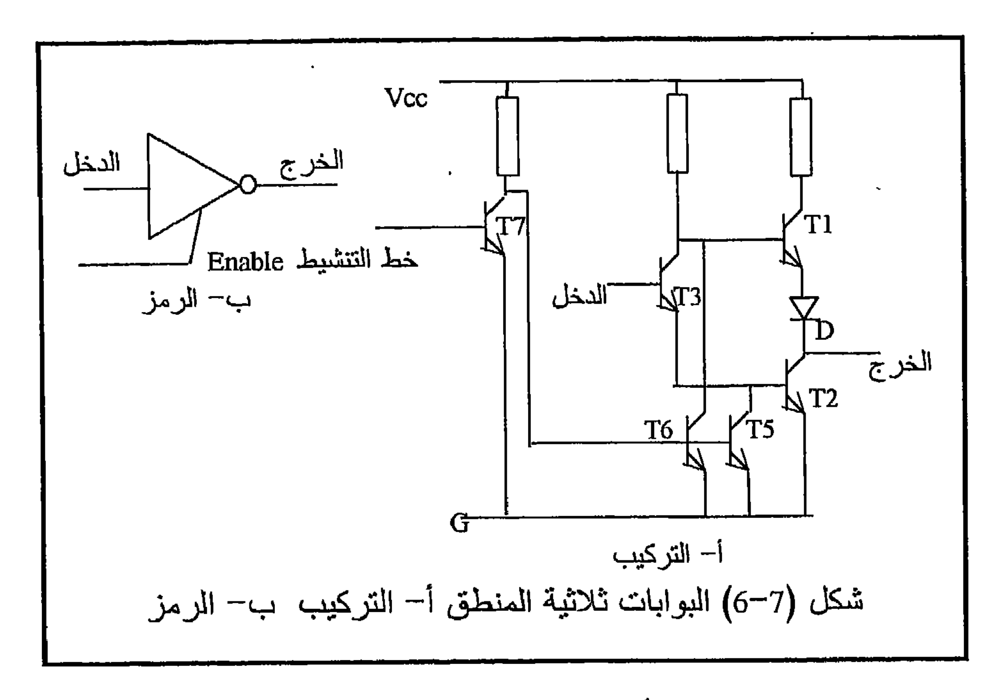
7-3-1 بوابات المنطق الثنائى Double state logic gates



شكل (7-5) بوابة عاكس NOT ثنائية المنطق أ- التركيب ب- الرمز

شكل (7-5) يبين تركيب ورمز بوابة من البوابات ثنائية المنطق وهى بوابة NOT . هذه البوابة دائما يكون خرجها إما واحدا أو صفرا على حسب حالة الدخل . إذا كان الدخل يساوى واحدا H فإن الترانزيستور T4 يكون مجمعه H وبالتالى فإن T3 يكون ON ويكون باعثه H مما يجعل T2 يكون ON وبالتالى فإن الخرج يكون موصلا إلى الأرضى أى يكون الخرج صفرا فى هذه الحالة. لاحظ أن T1 فى هذه الحالة يكون OFF أى غير موصل ولذلك فإن الخرج يكون معزولا من مصدر الجهد Vcc . إذا كان الدخل صفرا L فإن مجمع T4 يكون L وبالتالى T3 يكون OFF مما يجعل T2 يكون OFF وبالتالى T1 يكون ON وعلى ذلك فإن الخرج فى هذه الحالة يكون متصلا بمصدر الجهد Vcc ويكون واحدا أى H وفى نفس الوقت ينفصل عن الأرضى .

7-3-2 البوابات ثلاثية المنطق Tristate logic gate

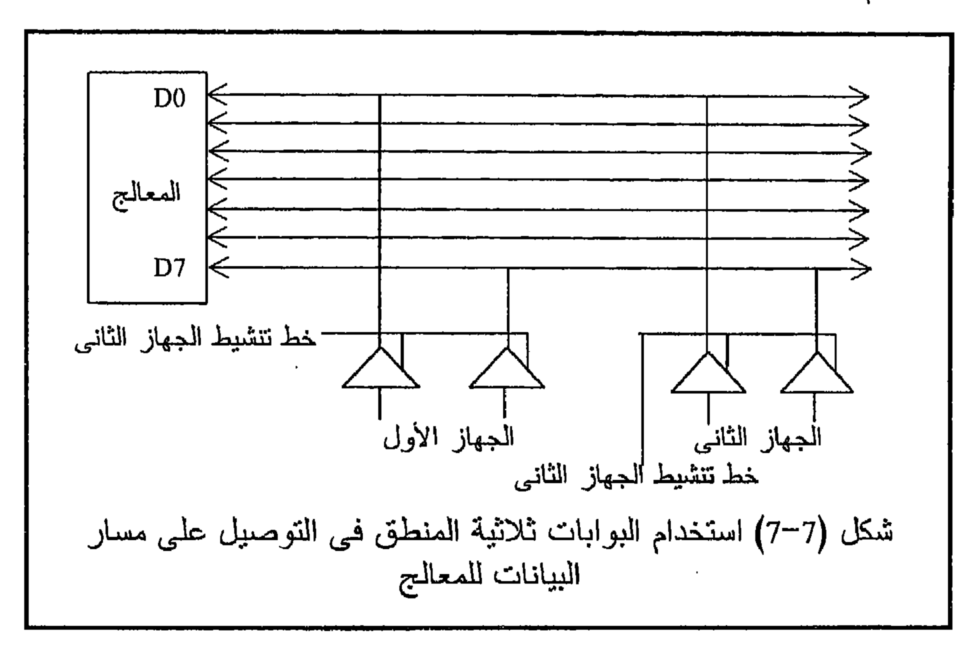


شكل (7-6) البوابات ثلاثية المنطق أ- التركيب ب- الرمز

تتميز البوابة ثلاثية المنطق أن لها طرفا ثالثا خاصا بالتحكم فى الخرج بحيث إذا كان هذا الطرف فعالا فإن البوابة ثلاثية المنطق تسلك نفس مسلك البوابة ثنائية المنطق تماما ، وأما إذا كان طرف التحكم غير فعال أو خامل فإن خرج البوابة ثلاثية المنطق يأخذ حالة جديدة غير معروفة فى البوابات ثنائية المنطق وهى أن الخرج لا يكون صفرا ولا واحدا وإنما يكون مفتوحا open circuit أو مقاومة عالية جدا high impedance .

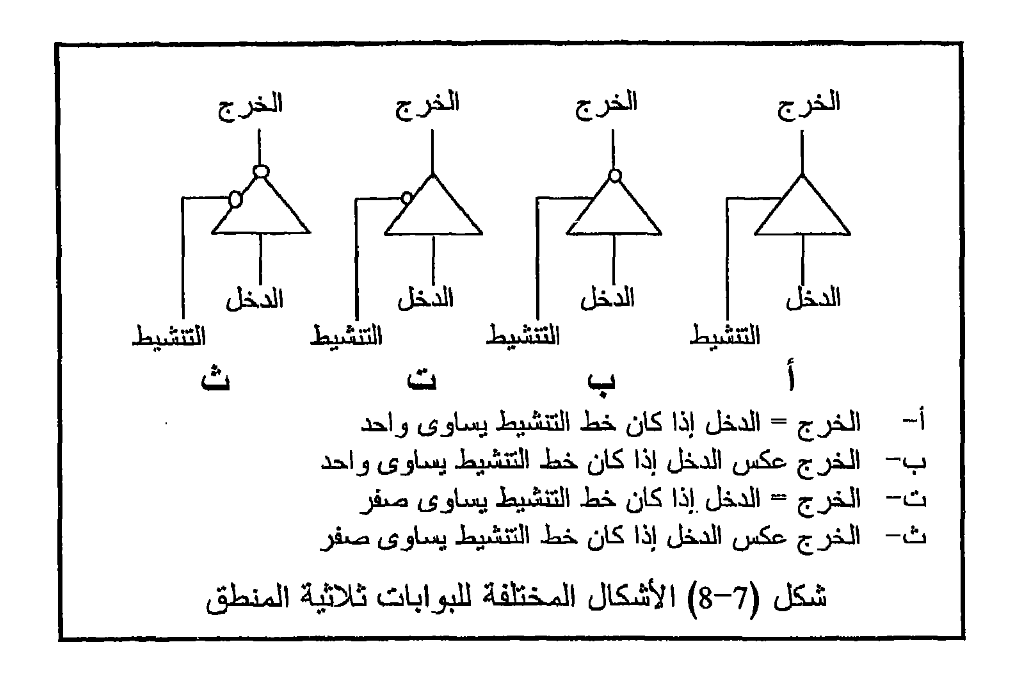
شكل (7-6) يبين الدائرة الإلكترونية والرمز المستخدم لمرحلة خرج بوابة ثلاثية المنطق وقد تم التحكم فيها عن طريق خط التنشيط enable . لاحظ أن مرحلة الخرج هذه هى نفسها مرحلة الخرج التى تم شرحها فى شكل (7-5) ولكن مضاف عليها الترانزيستورات T7, T6, T5 التى تعمل كمفاتيح يتم التحكم فيها عن طريق خط التنشيط enable . فإذا كان خط التنشيط عاليا H فإن T7 يكون ON مما يجعل T6, T5 كل منهما يكون OFF وبالتالى فإن البوابة تعمل كبوابة ثنائية المنطق مثل التى شرحت سابقا فى شكل (7-5) بحيث إذا كان الدخل H فإن الخرج يكون L والعكس . أما إذا كان خط التنشيط خاملا L فإن T7 يكون OFF مما سيجعل T6, T5 كل منهما يكون ON وبالتالى فإن كلا من T2, T1 يكون OFF وبالتالى فإن الخرج يكون غير موصل لا على الأرضى ولا على الجهد Vcc ولكن يكون كما لو كان مفتوحا open circuit أو مقاومة عالية.

السؤال الآن كيف ستستخدم البوابات ثلاثية المنطق فى الحماية من القصر الكهربى short circuits الذى يحدث بسبب توصيل أكثر من جهاز على نفس خطوط المسارات كما أوضحنا فى شكل (7-4) ؟
إن جميع الأجهزة التى ستوصل على مسار البيانات للمعالج يجب أن تكون مرحلة الخرج فيها عبارة عن بوابات ثلاثية المنطق وعن طريق خطوط التنشيط لكل جهاز فإن المعالج سيجعل جميع الأجهزة فى حالة خمول أى أن خرجها سيكون كما لو كان غير موصل على المسار إلا جهازا واحدا فقط وهو الجهاز الذى يتعامل معه المعالج فى تلك اللحظة . أى أنه نتيجة استخدام هذا النوع من البوابات فلن يكون هناك غير جهاز واحد فقط هو الفعال فى أى لحظة وهو الذى يتعامل معه المعالج وهو الذى سيكون موصلا على مسار البيانات وأما بقية الأجهزة فستكون منفصلة عن مسار البيانات نتيجة أن خط التنشيط الخاص بها غير فعال . شكل (7-7) يبين عملية توصيل أكثر من جهاز على مسار البيانات باستخدام البوابات ثلاثية المنطق .



شكل (7-7) استخدام البوابات ثلاثية المنطق فى التوصيل على مسار البيانات للمعالج

شكل (7-8) يبين الأنواع المختلفة للبوابات ثلاثية المنطق . هناك مثلا البوابات التى يكون خرجها مثل دخلها تماما إذا كان خط التنشيط فعالا ، كما أن هناك البوابات التى يكون خرجها عكس دخلها إذا كان خط التنشيط فعالا . هناك أيضا البوابات التى يكون خط تنشيطها فعالا عندما يكون صفرا ، وأخرى يكون خط تنشيطها فعالا عندما يكون واحد . مهمتك أنت كمصمم هى اختيار البوابة المناسبة للتطبيق الذى تستخدمه .



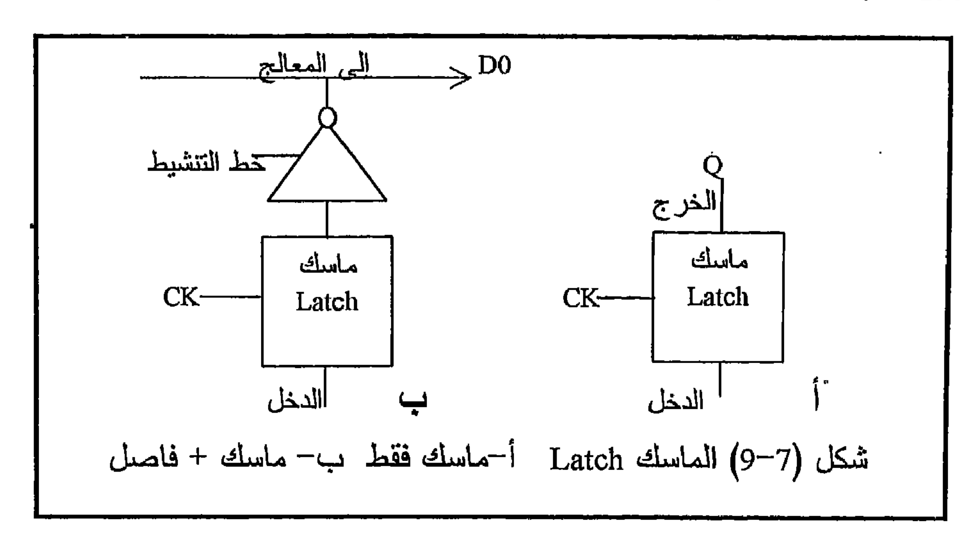
أ- الخرج = الدخل إذا كان خط التنشيط يساوى واحد
ب- الخرج عكس الدخل إذا كان خط التنشيط يساوى واحد
ت- الخرج = الدخل إذا كان خط التنشيط يساوى صفر
ث- الخرج عكس الدخل إذا كان خط التنشيط يساوى صفر

شكل (7-8) الأشكال المختلفة للبوابات ثلاثية المنطق

7-4 الماسك Latch

فى بعض التطبيقات تتطلب عملية الفصل استخدام مكونات لها خاصية مسك المعلومة على خرجها بالرغم من فقدان هذه المعلومة على الدخل . إن البوابات ثلاثية المنطق ليست لها هذه الخاصية لأنها طالما أن خط التنشيط فعال تكون هناك معلومات معينة على الخرج وبمجرد أن يكون خط التنشيط غير فعال فإن المعلومة التى على الخرج تفقد تماما . إن الماسك عبارة عن قلاب flip flop، وغالبا ما يكون من النوع D بحيث أن المعلومة التى على طرف الدخل D تنتقل إلى الخرج Q بعد وجود نبضة على طرف التزامن CK . تظل المعلومة الموجودة على الخرج كما هى لا تتغير حتى لو تغير الدخل D طالما أنه لم تعط أى نبضة تزامن أخرى ، لذلك فإننا نقول إن المعلومة قد مسكت على الخرج . شكل (7-9 أ) يبين مثل هذا الماسك . هناك أيضا الكثير من التطبيقات (أجهزة إدخال البيانات إلى المعالج مثلا) كما سنرى والتى تتطلب وجود بوابة ثلاثية المنطق بعد الماسك لتكون بمثابة فاصل بين مسار البيانات وخرج الماسك وذلك لأن الماسك وحده لا نستطيع توصيله على مسار البيانات مباشرة لأن خرجه يأخذ أحد الحالتين فقط إما الصفر أو الواحد أى ثنائى المنطق . ولذلك فإنه عادة يوضع بعد الماسك بوابة ثلاثية المنطق بحيث يوصل خرج الماسك على مسار البيانات فقط عندما يكون خط تنشيط البوابة ثلاثية المنطق فعالا . شكل (7-9ب)

يبين دائرة ماسك متبوعة ببوابة ثلاثية المنطق .



شكل (7-9) الماسك Latch أ-ماسك فقط ب- ماسك + فاصل

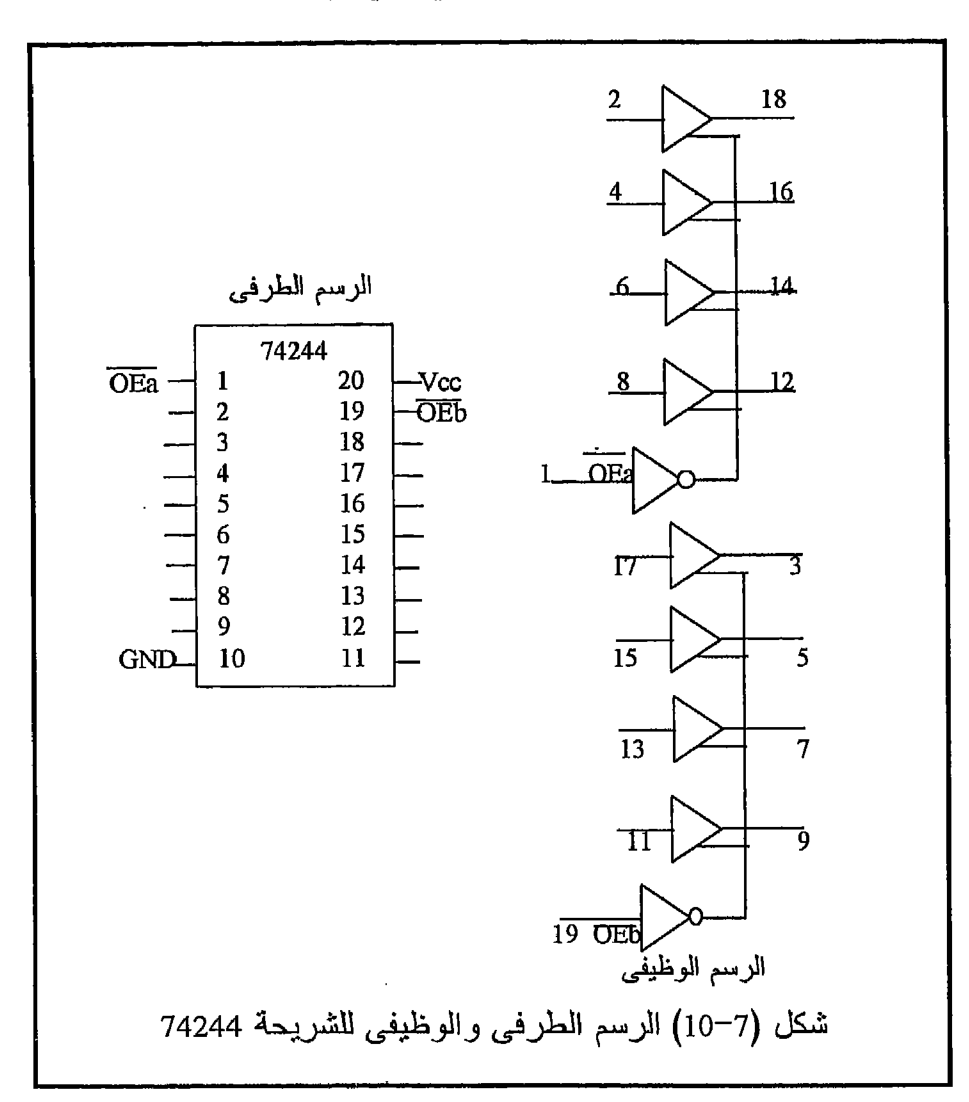
7-5 بعض الشرائح التى تستخدم فى فصل المسارات

سنعرض فى هذا الجزء لمكونات بعض الشرائح التى يمكن استخدامها فى عملية عزل المسارات ويجب أن نؤكد على أنه ليست هذه فقط هى الشرائح المتاحة ولكن هناك الكثير غيرها يمكن استخدامه ، فقط مطلوب قراءة كتالوجات هذه الشرائح وفهمها قبل البدء فى استخدامها .

7-5-1 الشريحة 74244 عازل ثمانى ثلاثى المنطق أحادى الاتجاه

بالنظر إلى الرسم الوظيفى والطرفى الموضحان فى شكل (7-10) يتضح لنا كيفية عمل هذه الشريحة . لاحظ أيضا أن هذه الشريحة أحادية الاتجاه ، أى أن الإشارة يمكن أن تمر فيها فى اتجاه واحد فقط على عكس الشريحة 74245 التى سيأتى شرحها . هذه الشريحة تحتوى على ثمانى بوابات ثلاثية المنطق مقسمة إلى مجموعتين ، المجموعة الأولى مكونة من 4 بوابات لها خط التنشيط الخاص بها على الطرف رقم 1 والمجموعة الثانية مكونة من الأربع بوابات الأخرى والتى لها خط تنشيط خاص بها هى الأخرى على الطرف رقم 19 . لاحظ أن خطوط التنشيط (1 و 19) تكون فعالة عندما تكون صفرا ، ولذلك فقد تم وضع شرطة فوق اسم كل منها ($\overline{OEa}$ و $\overline{OEb}$) . كما نعلم عن البوابات ثلاثية المنطق فإنه طالما أن خط التنشيط فعال (صفر فى هذه الحالة) فإن الموجود على دخل

هذه الشريحة ينتقل إلى خرجها . إن هذه الشريحة تصلح فقط لعزل خطوط مسار العناوين وذلك لأن الإشارة التى على هذا المسار تكون دائما خارجة من المعالج إلى الأجهزة المحيطة ولا تأخذ الاتجاه العكسى على الإطلاق .

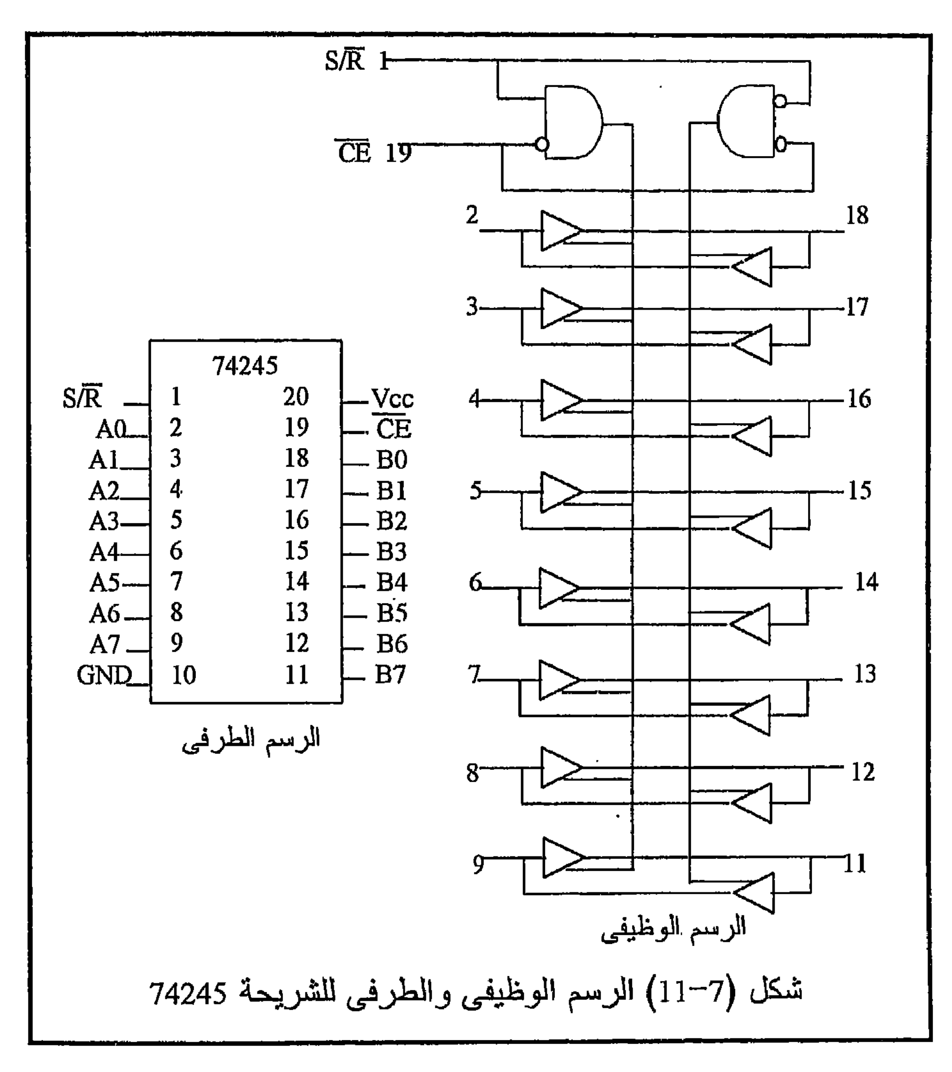


شكل (7-10) الرسم الطرفى والوظيفى للشريحة 74244

7-5-2 الشريحة 74245 عازل ثمانى ، ثلاثى المنطق ، ثنائى الاتجاه

هذه الشريحة عبارة عن عازل buffer ثنائى الاتجاه أى يمكنها إرسال واستقبال المعلومات ، وخرجها ثلاثى المنطق غير عاكس ولها خط تحكم فى الاتجاه وهو

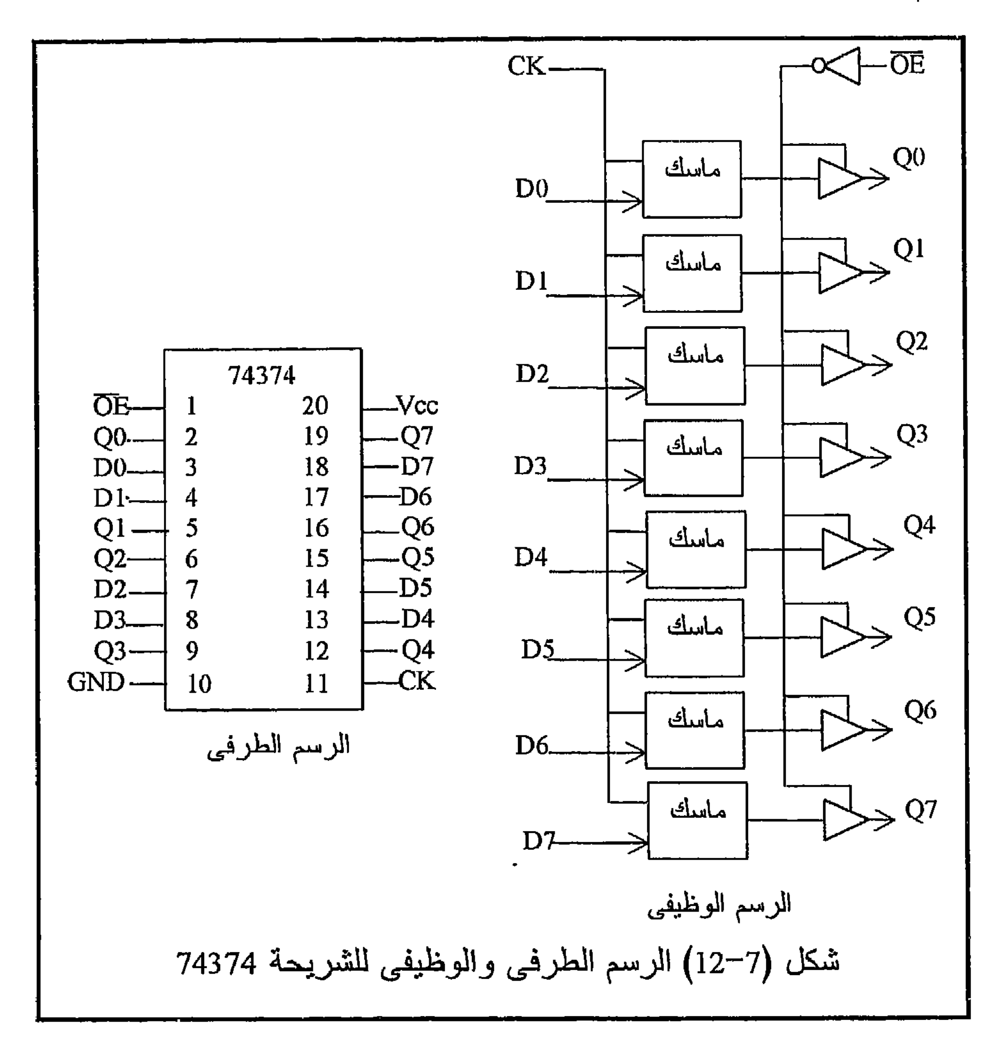
الطرف رقم 1 . الطرف رقم 19 هو خط تنشيط أو فعالية الشريحة الذى يكـــون فعال عندما يكون صفر Chip Enable, CE . شكل (7-11) يبين الرسم الطرفـــى والوظيفى للشريحة . عندما يكون الطرف 19 فعالا (صفر) فإن الشريحة تكـــون نشطة ، وعندما يكون هذا الطرف خاملا (واحد) فإن الشريحة تكون خاملـــة ولا تعمل . الطرف 1 يتحكم فى اتجاه البيانات خلال الشريحة . فعندما يكـــون هـــذا الطرف واحد فإن الإشارات تمر فى الاتجاه من A إلى B ، وأما إذا كـــان هـــذا الخط صفر فإن الإشارة تمر فى الاتجاه من B إلى A . لذلك فإن هذه الشـــريحة مناسبة جدا لفصل خطوط مسار البيانات كما سنرى .



شكل (7-11) الرسم الوظيفى والطرفى للشريحة 74245

7-5-3 الشريحة 74374 عازل ماسك ذو ثمانى بتات

تحتوى هذه الشريحة على ثمانية قلابات من النوع D كلها موصلة على نفس طرف التزامن CK وهو الطرف رقم 11 فى الشريحة . عند إعطاء نبضة تزامن على هذا الطرف تنتقل الإشارة الموجودة على جميع أطراف الدخل D إلى الخرج المناظر Q ، وتظل هذه الإشارة ممسوكة على الخرج طالما لم يتم إعطاء أى نبضات تزامن أخرى . كل واحد من هذه القلابات موصل على بوابة ثلاثية المنطق تسمح بمرور الإشارة إلى أطراف الخرج عندما يكون طرف التنشيط (الطرف 1) $\overline{OE}$ فعالا (صفر) . سنرى بعد قليل كيفية استخدام هذه الشريحة فى فصل مسار العناوين للمعالج 8085 بالذات لطبيعة مساراته . شكل (7-12) يبين الرسم الطرفى والوظيفى لهذه الشريحة .



شكل (7-12) الرسم الطرفى والوظيفى للشريحة 74374

الفصل الثامن

فصل مسارات المعالجات

Buffering of Microprocessor Buses

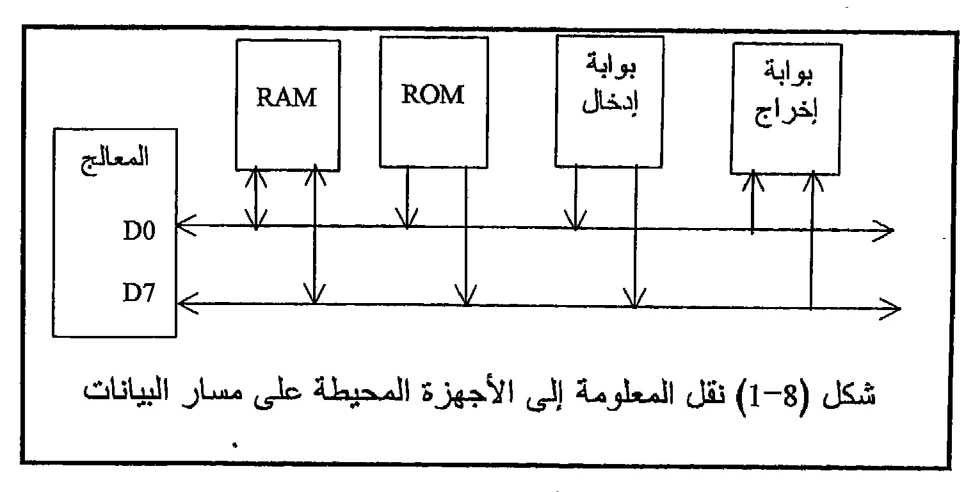
8-1 مقدمة

كما رأينا فى الفصل السابق فإن أى مسار bus يكون عبارة عن مجموعة من الخطوط المتوازية والتى عليها يمكن نقل معلومات أو إشارات من مكان لآخر . عادة تكون هذه المعلومات أو البيانات خارجة من مصدر معين وقاصدة إلى هدف آخر . بعض هذه المسارات يكون أحادى الاتجاه مثل مسار العناوين الذي دائما يحمل إشارات من المعالج إلى الأجهزة المحيطة ، والبعض الآخر يكون ثنائى الاتجاه مثل مسار البيانات الذى تكون عليه الإشارة خارجة من المعالج إلى الأجهزة المحيطة فى أزمنة معينة أو العكس من الأجهزة المحيطة إلى المعالج فى أزمنة أخرى . إن الهدف من عملية مواجهة المعالج مع الأجهزة المحيطة هو توفير الوسائل التى يستطيع بها المعالج التخاطب مع هذه الأجهزة ونعنى بكلمة التخاطب إرسال واستقبال معلومات أو إشارات إلى ومن هذه الأجهزة . شكل (8-1) يبين شريحة معالج وقد خرج منها مسار للبيانات إلى جميع الأجهزة المحيطة ، فهل هذا يكفى لحل جميع مشاكل عملية المواجهة ؟ كمثال على الأجهزة المحيطة نرى فى هذا الشكل بوابة إدخال وبوابة إخراج وذاكرة قراءة وكتابة RAM وذاكرة قراءة فقط ROM . لاحظ فى هذا الشكل أن المعالج كما لو كان بنكا أو دكانا للمعلومات وجميع الأجهزة المحيطة تريد التعامل معه من خلال مسار البيانات .

عند مواجهة (توصيل) المعالج مع أى جهاز من الأجهزة المحيطة تنشأ مشكلتان يجب التغلب عليهما وهما كما يلى :

أولا: يجب التأكد من أنه فى أى لحظة لا يتم نقل أى معلومة إلا لجهاز واحد فقط ، أى أنه عندما يكون المعالج فى حالة اتصال (مخاطبة) مع أى جهاز من الأجهزة المحيطة فإنه يكون على اتصال بهذا الجهاز فقط دون الأجهزة الأخرى. فمثلا نريد أن نضمن أنه عندما سيرسل المعالج معلومة إلى أى بوابة إخراج فإن هذه المعلومة لن تذهب أيضا إلى أى بايت من بايتات ال RAM .

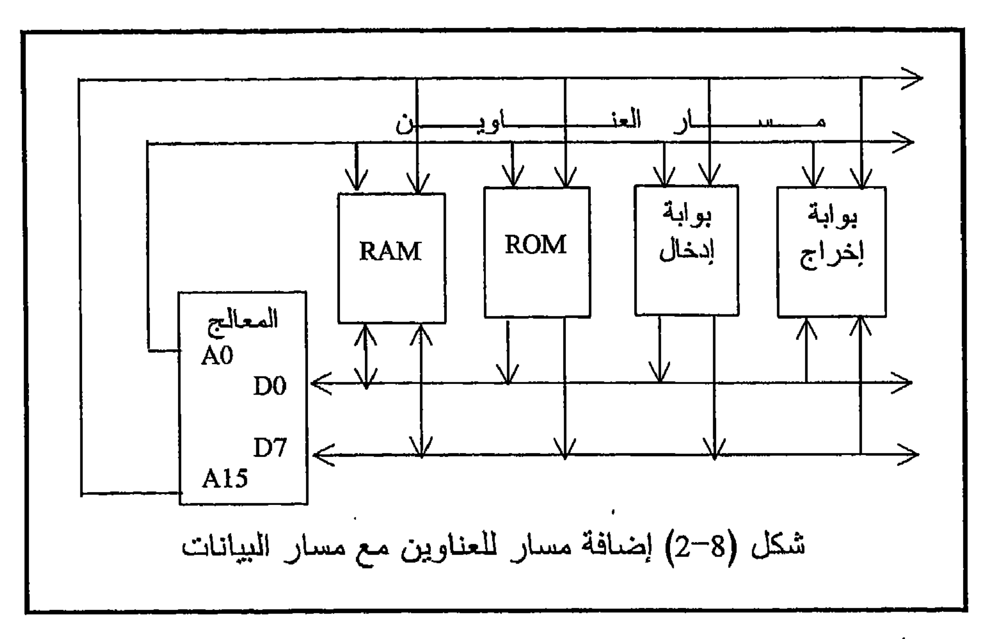
ثانيا: المشكلة الثانية هى أنه يجب التأكد من أنه عند اتصال المعالج بأى واحد من الأجهزة المحيطة فإن الأجهزة الأخرى لن تشوش أو تتداخل فى عملية الاتصال . فمثلا عندما يريد المعالج أن يقرأ معلومة من ال RAM فإننا نريد أن نضمن أن ال ROM أو أى بوابة إدخال لن تتدخل وترسل هى الأخرى معلومات إلى المعالج بحيث يحدث فى هذه الحالة ما يسمى بتصادم للمعلومات على مسار البيانات وقد أشرنا لذلك فى الفصل السابق فى معرض حديثنا عن البوابات ثلاثية المنطق .



شكل (8-1) نقل المعلومة إلى الأجهزة المحيطة على مسار البيانات

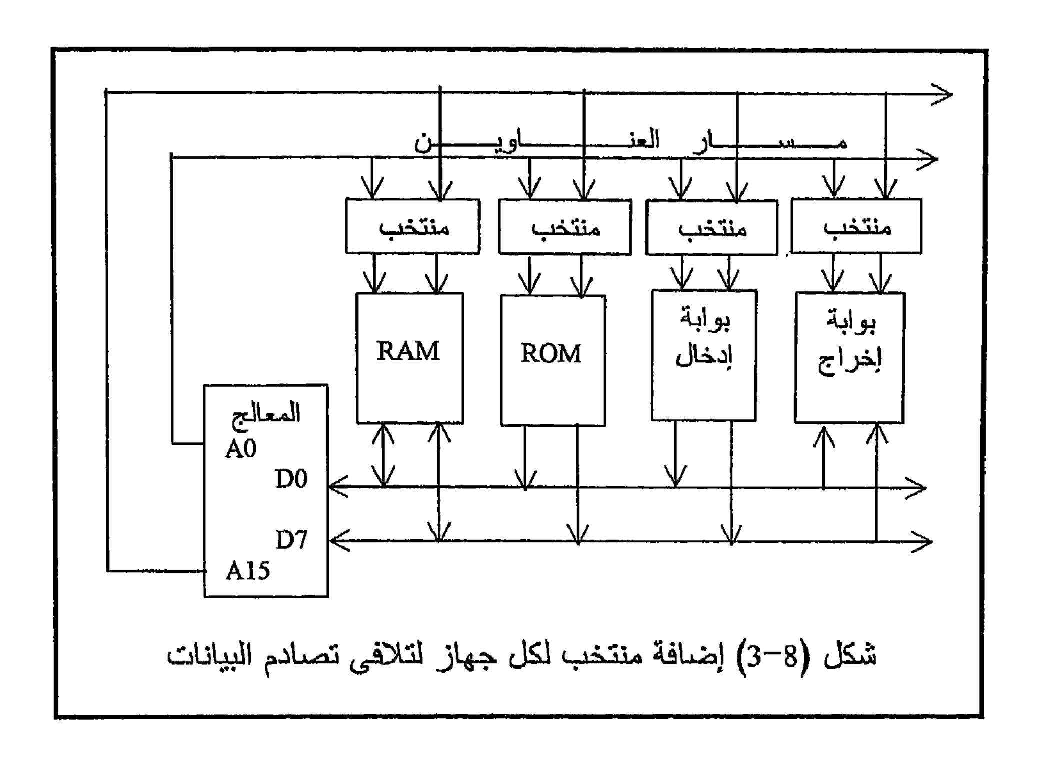
8-2 لماذا مسار العناوين ؟

إن المشكلة الأولى وهى مشكلة ضمان الاتصال أو التعامل مع جهاز واحد فقط يمكن التغلب عليها عن طريق استخدام مسار للعناوين بحيث يكون لكل جهاز من الأجهزة المحيطة عنوانا خاصا به يتم إخراجه على مسار العناوين من المعالج ولا يتعرف على هذا العنوان إلا الجهاز المعنى به فقط فيصبح هذا الجهاز (الذى تعرف على عنوانه) فى حالة نشاط أو فعالية فيستقبل المعلومة الموجودة على مسار البيانات . وأما جميع الأجهزة الأخرى التى لم تتعرف على العنوان فإنها تكون خاملة ويتم منعها من التعامل مع مسار البيانات ، لذلك ظهرت الحاجة إلى مسار للعناوين بجانب مسار البيانات . شكل (8-2) يبين مسار العناوين وقد أضيف إلى النظام الموجود فى شكل (8-1) . عادة يتكون مسار العناوين من عدد معين من الخطوط ويتم تحديد هذا العدد عن طريق مصمم (صانع) شريحة المعالج . هذا المسار سيحمل إشارة ثنائية (وحايد وأصفار) وكل إشارة تمثل شفرة أو كودا لعنوان واحد من الأجهزة المحيطة التى يستطيع المعالج التعامل معها . لذلك فإن عدد هذه الأجهزة يساوى اثنين أس عدد الخطوط الموجودة فى مسار العناوين ، فلو كان مثلا عدد خطوط مسار العناوين يساوى ستة خطوط فإن عدد الأجهزة سيكون 64 جهازا (2^6) . إن بعض شرائح المعالجات التى سنتعامل معها فى هذا الكتاب يحتوى مسار العناوين فيها على 16 خطا أو 16 بت ، ولذلك فإنها تستطيع التعامل مع 2^{16} أى 65536 جهازا أو عنوانا من الأجهزة المحيطة بين بوابات إدخال وبوابات إخراج وبايتات ذاكرة كل منها له العنوان الخاص به والمكون من 16 بت .



شكل (8-2) إضافة مسار للعناوين مع مسار البيانات

كما رأينا فإن نظام العنونة (كما فى شكل (8-2) والذى سيأتى تفصيله فيما بعد) قد حل المشكلة الأولى وهى مشكلة ضمان عدم تعامل المعالج مع أكثر من واحد من الأجهزة المحيطة . أما المشكلة الثانية وهى عدم التداخل بين الأجهزة المحيطة على مسار البيانات أو عدم تصادم المعلومات على نفس المسار فهذه قد أوضحنا فى الفصل السابق أن سببها يرجع إلى استخدام البوابات ثنائية المنطق فى مراحل خرج الأجهزة المحيطة ، ولقد أوضحنا فى الفصل السابق أيضا أن حل هذه المشكلة يكون عن طريق استخدام البوابات ثلاثية المنطق فى مراحل خرج هذه الأجهزة بحيث عندما يريد المعالج التعامل مع أى جهاز فإنه يقوم بتنشيط خط التحكم فى مرحلة خرج هذا الجهاز فقط وأما باقى الأجهزة الموصلة على مسار البيانات فتكون خاملة أو كما لو كانت غير موصلة على مسار البيانات open circuit . شكل (8-3) يبين عملية توصيل الأجهزة المحيطة مع المعالج وقد استخدم منتخب أو فاكك أو محلل شفرة ملحق بكل جهاز بحيث يصبح خرج هذا المحلل فعالا إذا كان العنوان الموجود على مسار العناوين مطابقا تماما للعنوان الذى صمم من أجله هذا المحلل ، حيث فى هذه الحالة يكون هذا هو الجهاز الوحيد الذى سيتعامل مع المعالج . إن عملية تصميم منتخب أو محلل شفرة لكل جهاز من الأجهزة المحيطة سنتعرف عليها بالتفصيل فى الفصول القادمة فى معرض الحديث عن مواجهة الذاكرة وبوابات الإدخال والإخراج .



شكل (3-8) إضافة منتخب لكل جهاز لتلافى تصادم البيانات

3-8 لماذا مسار التحكم ؟

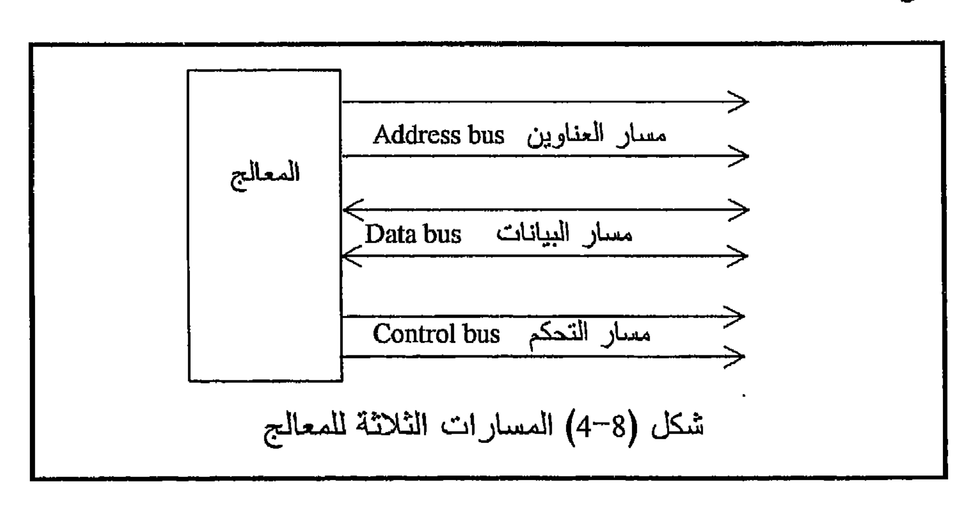
افترض مثلا أن المعالج يريد كتابة (إرسال) المعلومة أو الرقم 55H إلى البايت التى عنوانها E100H ، فماذا سيفعل المعالج على ضوء معرفتنا بوظيفة كل من مسار العناوين والبيانات ؟ إن المعالج لكى يقوم بهذه المهمة فإنه سيضع العنوان E100H على مسار العناوين وبذلك تصبح شريحة الذاكرة التى تحتوى هذه البايت نشطة وعلى استعداد للتعامل مع المعالج ، عند ذلك يقوم المعالج بوضع المعلومة 55H على مسار البيانات فتتلقاها البايت المعنية وتسجل فيها . المشكلة هنا هى أن المعالج عندما قام بتنشيط شريحة الذاكرة التى تحتوى هذه البايت لم يخبر الشريحة عما إذا كان سيرسل إليها معلومات أم سيستقبل منها ، أى هل سيكتب فيها أم سيقرأ منها . لذلك كان من الضرورى أن يكون هناك خط تحكم يخرج من المعالج يخبر الجهاز الذى سيتعامل معه المعالج عن الهدف من هذا التعامل ، هل هو بغرض القراءة أم بغرض الكتابة . مثل هذا الخط وخطوط أخرى تجمع تحت اسم مسار التحكم control bus وعدد الخطوط فى هذا المسار يختلف من معالج لآخر . سنذكر هنا أهم أربعة خطوط تحكم وسنترك الباقى

للكلام عنه فى مواضع استخدامه .

من خطوط التحكم ما يلى :

1. خط قراءة الذاكرة memory read, MEMR وهذا الخط يقوم المعالج بتنشيطه فى حالة القراءة من الذاكرة (RAM أو ROM) .
2. خط الكتابة فى الذاكرة memory write, MEMW وهذا الخط يقوم المعالج بتنشيطه فى حالة الكتابة فى الذاكرة (RAM) .
3. خط قراءة بوابة إدخال input port read, IOR وهذا الخط يكون فعالا عندما يكون المعالج فى حالة استقبال معلومات من بوابة إدخال .
4. خط كتابة فى بوابة إخراج output port write, IOW وهذا الخط يكون فعالا عندما يكون المعالج فى حالة إرسال للمعلومات إلى بوابة إخراج .

لاحظ أن واحد فقط من هذه الخطوط (خطوط التحكم) يكون فعالا فى أى لحظة وباقى الخطوط تكون خاملة ولذلك فإن تسمية هذه المجموعة من الخطوط بالمسار تعتبر تسمية مجازية ومن الصواب أن تسمى خطوط التحكم فقط ولكن جرى العرف على إطلاق اسم مسار التحكم عليها . شكل (8-4) يبين شريحة معالج وقد خرج منها المسارات الثلاثة : العناوين والبيانات والتحكم . لاحظ أن عدد خطوط مسار العناوين 16 خطا فى بعض المعالجات التى ندرسها فى هذا الكتاب (وهى المعالجات ذات 8 بت) وسيصل عدد خطوطه إلى 32 بت كما سنرى فى المعالجات الحديثة . وكذلك عدد خطوط مسار البيانات 8 خطوط فى المعالجات ذات 8 بت وسيصل إلى 32 أيضا كما سنرى . أما عدد خطوط مسار التحكم فلم يكتب فى شكل (8-4) لأن هذا العدد كما أشرنا يختلف من معالج لآخر .



شكل (8-4) المسارات الثلاثة للمعالج

8-4 تهيئة مسارات المعالج 8085 لعملية المواجهة

		المعالج 8085		
X1	1		40	Vcc
X2	2		39	HOLD
Reset out	3		38	HLDA
SOD	4		37	CLK OUT
SID	5		36	$\overline{\text{Reset in}}$
TRAP	6		35	READY
RST7.5	7		34	IO/M
RST6.5	8		33	S1
RST5.5	9		32	$\overline{RD}$
INTR	10		31	$\overline{WR}$
$\overline{INTA}$	11		30	ALE
AD0	12		29	S0
AD1	13		28	A15
AD2	14		27	A14
AD3	15		26	A13
AD4	16		25	A12
AD5	17		24	A11
AD6	18		23	A10
AD7	19		22	A9
Vss	20		21	A8

شكل (8-5) أطراف الشريحة 8085

إذا ألقينا نظرة فاحصة على أطراف الشريحة 8085 كما فى شكل (8-5) فى محاولة للتعرف على المسارات المختلفة لهذا المعالج لوجدنا الآتى :

1. مسار العناوين يمكن التعرف على 8 خطوط فقط منه وهى الخطوط A8 إلى A15 أما باقى الخطوط فليست موجودة بالصورة المباشرة .
2. مسار البيانات أيضا من الصعب التعرف عليه بالصورة المباشرة .
3. ماذا تعنى الخطوط AD0 إلى AD7 هل هى خطوط لمسار العناوين أم لمسار البيانات؟
4. خطوط التحكم كما عرفناها مسبقا ليست موجودة أيضا بالصورة المباشرة ، ولكن الموجود هو الخطوط $\overline{RD}$ و $\overline{WR}$ فهل هذه الخطوط لها علاقة بخطوط القراءة من الذاكرة $\overline{MEMR}$ والكتابة فى الذاكرة $\overline{MEMW}$ التى تكلمنا عنها فى معرض الحديث عن مسار التحكم ؟

جميع هذه الأسئلة وزيادة سنجيب عنها فى هذا الجزء فى محاولة للحصول على

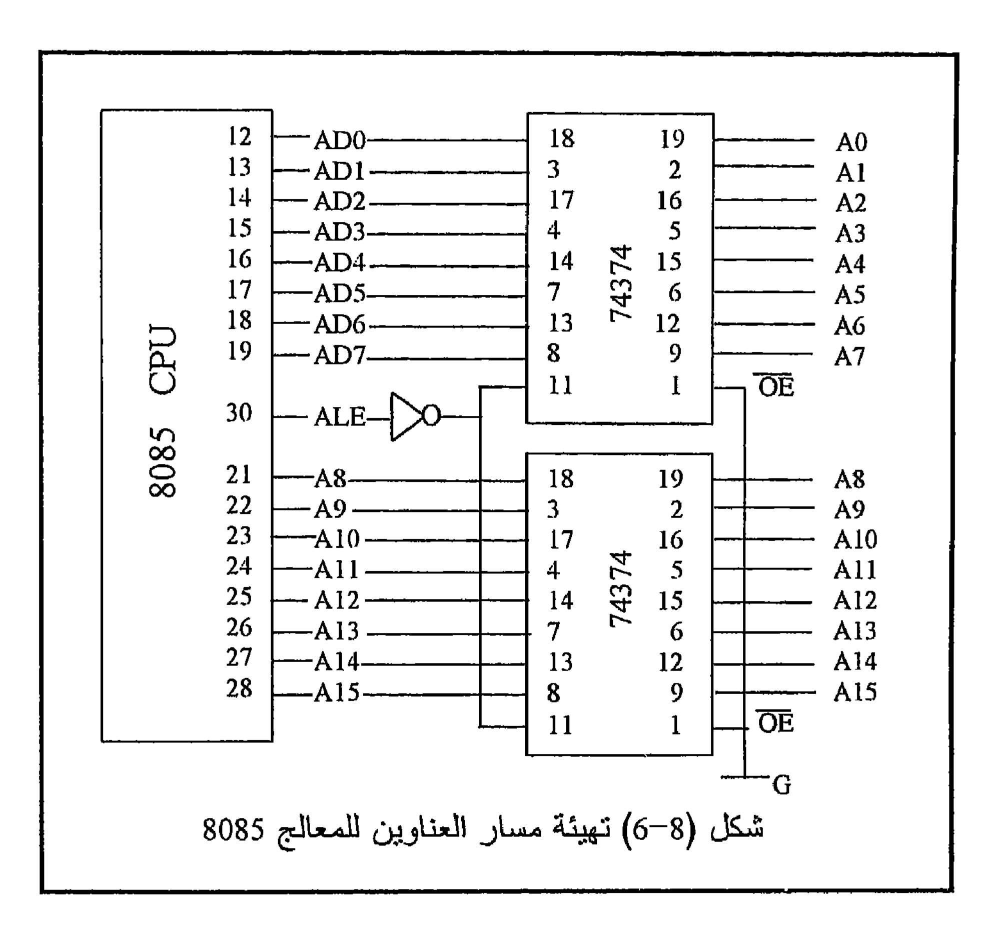
المسارات الثلاثة فى الصورة المباشرة والملائمة لعملية توصيل هذا المعالج مع الأجهزة المحيطة .

8-4-1 مسار العناوين للمعالج 8085

تختلف الشريحة 8085 عن الكثير من الشرائح من حيث أن كل من مسار العناوين والبيانات يستخدمان نفس الخطوط AD0 إلى AD7 فى عملية مشاركة زمنية time multiplexing بحيث أن الإشارة الموجودة على هذه الخطوط تكون إشارة عناوين فى بداية كل دورة أمر ثم تكون بعد ذلك إشارة بيانات . أى أن الإشارة الموجودة على الخطوط AD0 إلى AD7 تمثل عنوان للحظة وجيزة فى بداية كل دورة أمر ثم تختفى إشارة العنوان وتصبح الإشارة الموجودة هى إشارة بيانات ، ولذلك فإننا لو استطعنا مسك إشارة العناوين أثناء هذه اللحظة على ماسك لحصلنا على العنوان بالكامل A0 إلى A15 . السؤال هنا هو: هل هناك وسيلة لمعرفة متى تكون الإشارة على هذه الخطوط الثمانية AD0 إلى AD7 تمثل عناوين ومتى تمثل بيانات ؟ لقد أجاب البروسيسور 8085 على هذا السؤال وأعطانا الخط ALE على الطرف 30 والذى عن طريقه يمكن معرفة نوع الإشارة على الخطوط AD0-AD7 . إن الحروف ALE تعنى Address Latch Enable أى منشط ماسك العناوين . هذا الخط يكون واحد عندما تكون الإشارة على الخطوط AD0-AD7 تمثل عناوين ، ويكون صفرا عندما تكون الإشارة على هذه الخطوط تمثل بيانات . بذلك نستطيع استخدام هذا الخط كخط تحكم أو خط تنشيط لشريحة ماسك تقوم بمسك أو تخزين الإشارة على الخطوط AD0-AD7 عندما يكون الخط ALE يساوى واحدا وبذلك نكون قد حصلنا على العنوان بالكامل A0-A15 . شكل (8-6) يبين الخطوط AD0-AD7 وقد أدخلت على الشريحة 74374 التى هى عبارة عن ماسك ثمانى كما شرحناها فى الفصل السابق . ولقد تم توصيل الطرف ALE من المعالج إلى طرف التزامن clock للشريحة 74374 من خلال عاكس NOT حتى نضمن أن عملية مسك العنوان ستتم عند نزول الخط ALE من الواحد إلى الصفر بناء على طلب المعالج . فى الشكل (8-6) نلاحظ أن الخطوط A8-A15 قد أدخلت هى الأخرى على شريحة ماسك مثل الخطوط AD0-AD7 فهل هناك ضرورة لذلك ؟

فى الحقيقة ليست هناك ضرورة لذلك ولكننا استخدمنا الشريحة 74374 فى هذا المكان لتحقيق عملية فصل buffer لهذه الخطوط حتى تستطيع إمداد جميع الدوائر المحيطة بالتيارات اللازمة . لاحظ أن الطرف رقم 1 ($\overline{OE}$) فى الشريحة رقم 74374 وهو خط التحكم فى البوابات ثلاثية المنطق الموجودة فى مرحلة خرج هذه الشريحة قد تم توصيله على الأرضى حتى تكون مرحلة الخرج فعالة دائما ، أى أنه بمجرد مسك العنوان فإنه يصبح مباشرة موجودا على خرج

الشريحة . يمكن توصيل هذا الخط بالطرف HOLD القادم من المعـالج لوضـع مسار العناوين فى حالة المقاومة العالية عند اللزوم . شكل (8-7) يبين الـتزامن الموجود بين الإشارة ALE والإشارات الموجودة علـى الخطـوط AD0-AD7 وكذلك الخطوط AD8-AD15 . لاحظ أهمية تخزين محتويات الخطـوط AD0-AD7 عند الحافة الهابطة لخط التحكم ALE .

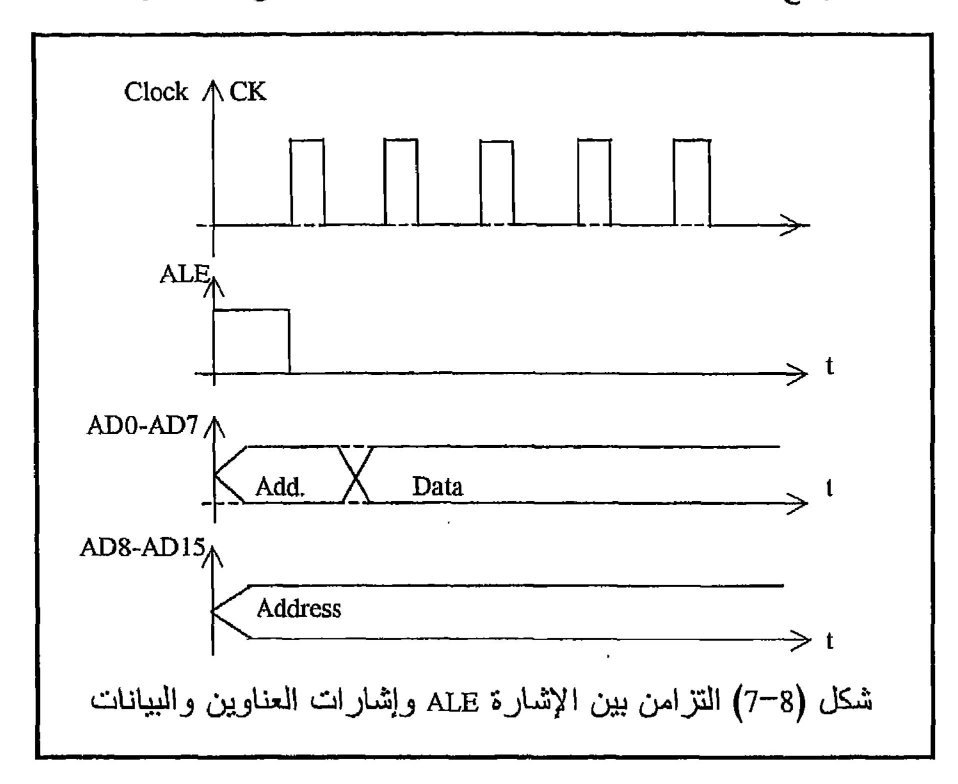


شكل (8-6) تهيئة مسار العناوين للمعالج 8085

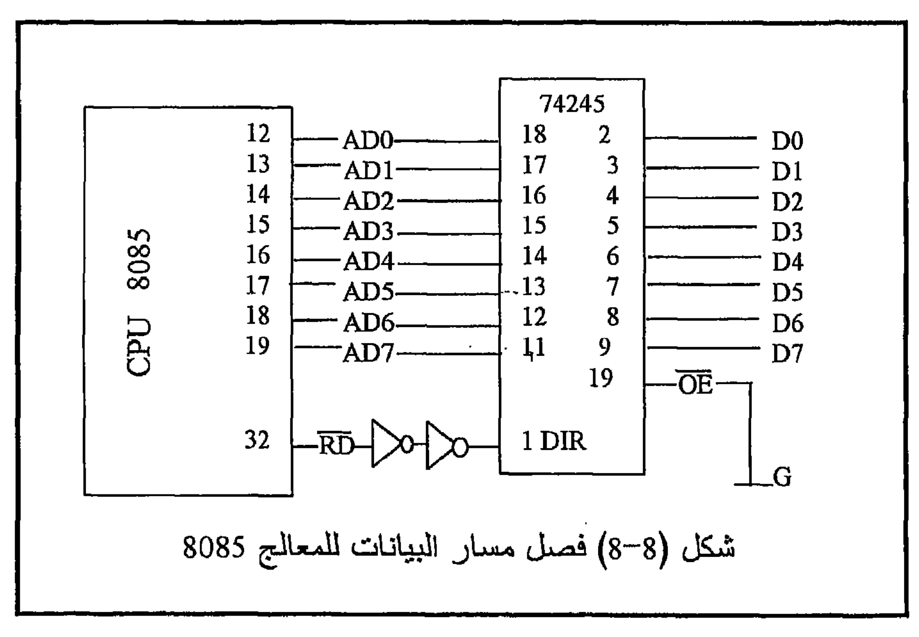
8-4-2 مسار البيانات للشريحة 8085

الآن وقد تم فصل مسار العناوين وتهيئته فإن الإشارة الموجودة علـى الخطـوط AD0-AD7 تمثل إشارة بيانات فى الزمن المتبقى من دورة الأمر ، والمطلـوب فقط هو عملية فصل buffer لمسار البيانات حتـى يسـتطيع توفـير التيـارات اللازمة للأجهزة المحيطة الموصلة عليه . لاحظ أن مسار البيانات ثنائى الاتجـاه لذلك يجب مراعاة استخدام الشريحة المناسبة له وقد أوضحنا فى الفصل السـابق

أنه من الشرائح المرشحة لهذه العملية الشريحة 74245 والتى سبق شرحها .



شكل (8-7) التزامن بين الإشارة ALE وإشارات العناوين والبيانات



شكل (8-8) فصل مسار البيانات للمعالج 8085

شكل (8-8) يبين عملية فصل أو تهيئة مسار البيانات . نلاحظ فى هذا الشـــكل أن الطرف رقم 1 فى الشريحة 74245 هو طرف التحكم فى اتجاه الإشارة $\overline{DIR}$ ولقد تم توصيل هذا الطرف بالطرف رقم 32 فى المعالج وهو طرف القـــراءة $\overline{RD}$ ، بحيث عندما يكون هذا الطرف ($\overline{RD}$) فعالا أى يساوى صفـــرا فـــإن الشـــريحة 74245 ستسمح بمرور البيانات من الأجهزة المحيطة إلى المعالج . بينما إذا كـــان الطرف $\overline{RD}$ غير فعال ، أى يساوى واحد ، فإن الشريحة 74245 ستسمح بمرور البيانات من المعالج إلى الأجهزة المحيطة . نلاحظ أيضا مـــن شـــكل (8-8) أن الخط $\overline{RD}$ قبل توصيله إلى الشريحة 74245 تم فصله عن طريق توصيلـــه مـــن خلال عاكسين .

8-4-3 مسار التحكم للشريحة 8085

إن مسار التحكم المبسط يتكون كما ذكرنا سابقا من 4 خطوط فقط وهى كالتالى:

- قراءة من الذاكرة Memory read
- كتابة فى الذاكرة Memory write
- قراءة من جهاز إدخال Input device read
- كتابة فى جهاز إخراج Output device write

طرف 32 $\overline{RD}$	طرف 31 $\overline{WR}$	طرف 34 $IO/\overline{M}$	
0	1	1	$\overline{IOR}$
1	0	1	$\overline{IOW}$
0	1	0	$\overline{MEMR}$
1	0	0	$\overline{MEMW}$

شكل (8-9) جدول حقيقة للحصول على خطوط التحكم الأربعة للمعالج 8085

جميع هذه الخطوط فعالة عندما تكون صفرا active low ولو فحصنـــا أطـــراف الشريحة 8085 فإننا لن نجد أن هذه الخطوط الأربعة بالصورة المباشـــرة التـــى نريدها ولكننا سنجد ثلاثة خطوط فقـــط وهـــى الخطـــوط $\overline{RD}$ و $\overline{WR}$ و $IO/\overline{M}$ والمطلوب هو الحصول على خطوط التحكم الأربعة السابقة من هـــذه الخطـــوط الثلاثة . إن السر يكمن فى الطرف $IO/\overline{M}$ حيث أن هذا الخط يكون واحدا عندمـــا يكون المعالج يتعامل مع أجهزة إدخال أو إخراج أى ينفذ واحد من الأمريـــن IN أو OUT ، كما أن الخط $IO/\overline{M}$ يكون صفرا فى حالة ما إذا كان المعالج يتعـــامل مع ذاكرة . ولذلك إذا كان الخط $\overline{RD}$ فعالا (1) ، وكـــان الخـــط $IO/\overline{M}$ يســـاوى صفرا ، فإن ذلك يعنى أن المعالج فى حالة قراءة من الذاكرة . أما إذا كان الخـــط

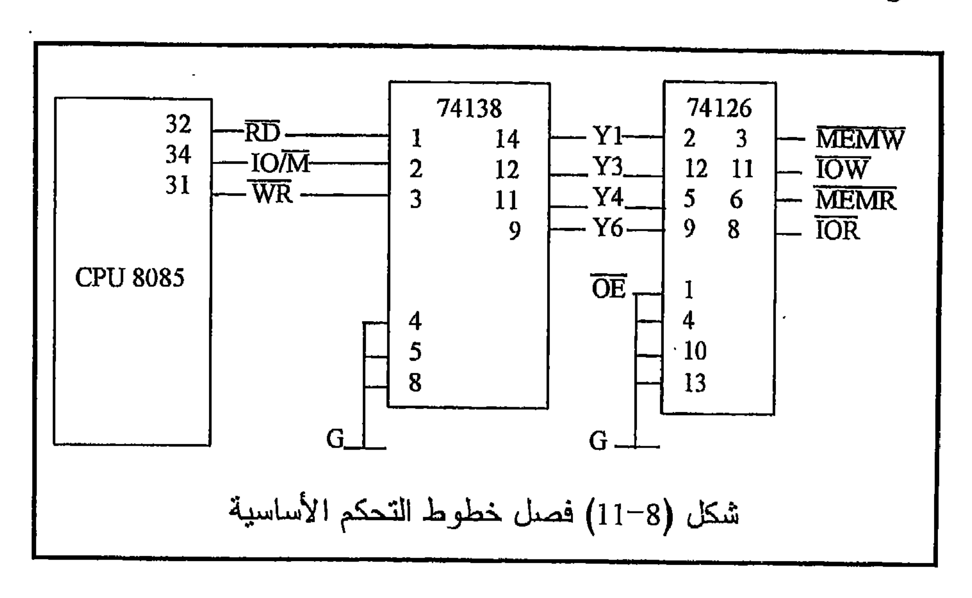
$\overline{RD}$ يساوى صفر ، والخط $IO/\overline{M}$ يساوى واحد ، فإن ذلك يعنى أن المعالج فـــى حالة قراءة من جهاز إدخال . شكل (8. 9) يبين جدول الحقيقة لجميع الخطـــوط الأربعة المطلوبة وحالة كل خط من الخطوط الثلاثة $\overline{RD}$ و $\overline{WR}$ و $IO/\overline{M}$.
بعد دراسة جدول الحقيقة المبين فى شكل (8. 9) يمكن بناء أكثر من دائرة يكـون دخلها هو الخطوط $\overline{RD}$ و $\overline{WR}$ و $IO/\overline{M}$ وخرجها هو الخطوط الأربعـة $\overline{MEMR}$ و $\overline{MEMW}$ و $\overline{IOR}$ و $\overline{IOW}$.
سنرى هنا طريقة الحصول على هذه الخطـــوط باسـتخدام المنتخـب أو فـاكك الشفرة Decoder رقم 74138 وهو عبارة عن منتخب لخط واحد مـــن خطـوط الخرج الثمانية Y0 إلى Y7 وهذا الانتخاب يكون على حسب شفرة توضع علـــى خطوط الدخل الثلاثة A, B, C . شكل (8–10) يبين جدول الحقيقة لهذا المنتخــب وقد تم توصيل دخوله الثلاثة على الخطوط $\overline{RD}$ و $\overline{WR}$ و $IO/\overline{M}$.

INPUT الـدخل			OUTPUT الـــخرج							
$\overline{WR}$	$IO/\overline{M}$	$\overline{RD}$		$\overline{MEMW}$		$\overline{IOW}$	$\overline{MEMR}$		$\overline{IOR}$	
C	B	A	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
L	L	L	L	H	H	H	H	H	H	H
L	L	H	H	L	H	H	H	H	H	H
L	H	L	H	H	L	H	H	H	H	H
L	H	H	H	H	H	L	H	H	H	H
H	L	L	H	H	H	H	L	H	H	H
H	L	H	H	H	H	H	H	L	H	H
H	H	L	H	H	H	H	H	H	L	H
H	H	H	H	H	H	H	H	H	H	L

شكل (8–10) جدول الحقيقة للشريحة 74138

طبقا لهذا الشكل فإن الخط $\overline{MEMR}$ سيؤخذ من علـــى الخــرج Y4 للمنتخــب ، والخط $\overline{MEMW}$ يؤخذ من على الخرج Y1 ، وأما الخط $\overline{IOR}$ فيؤخذ من علـــى الخرج Y6 ، والخط $\overline{IOW}$ يؤخذ من على الخـــرج Y3 . وأمـــا بـــاقى خـــروج المنتخب فإنها غير مستخدمة . شكل (8. 11) يبين كيفية توصيل هذا المنتخــــب مع المعالج . لاحظ من هذا الشكل أن خطوط الخرج الأربعة تم توصيلها علـــى الشريحة 74125 وهى فاصل buffer ثلاثى المنطق وقد وصلت جميـــع خطـوط تنشيط البوابات بالأرضى حتى تكون هذه الخطوط فى حالة نشاط دائـم . يمكـن عند الضرورة توصيل خطوط التنشيط هذه بالخط HOLD القادم من المعالج .

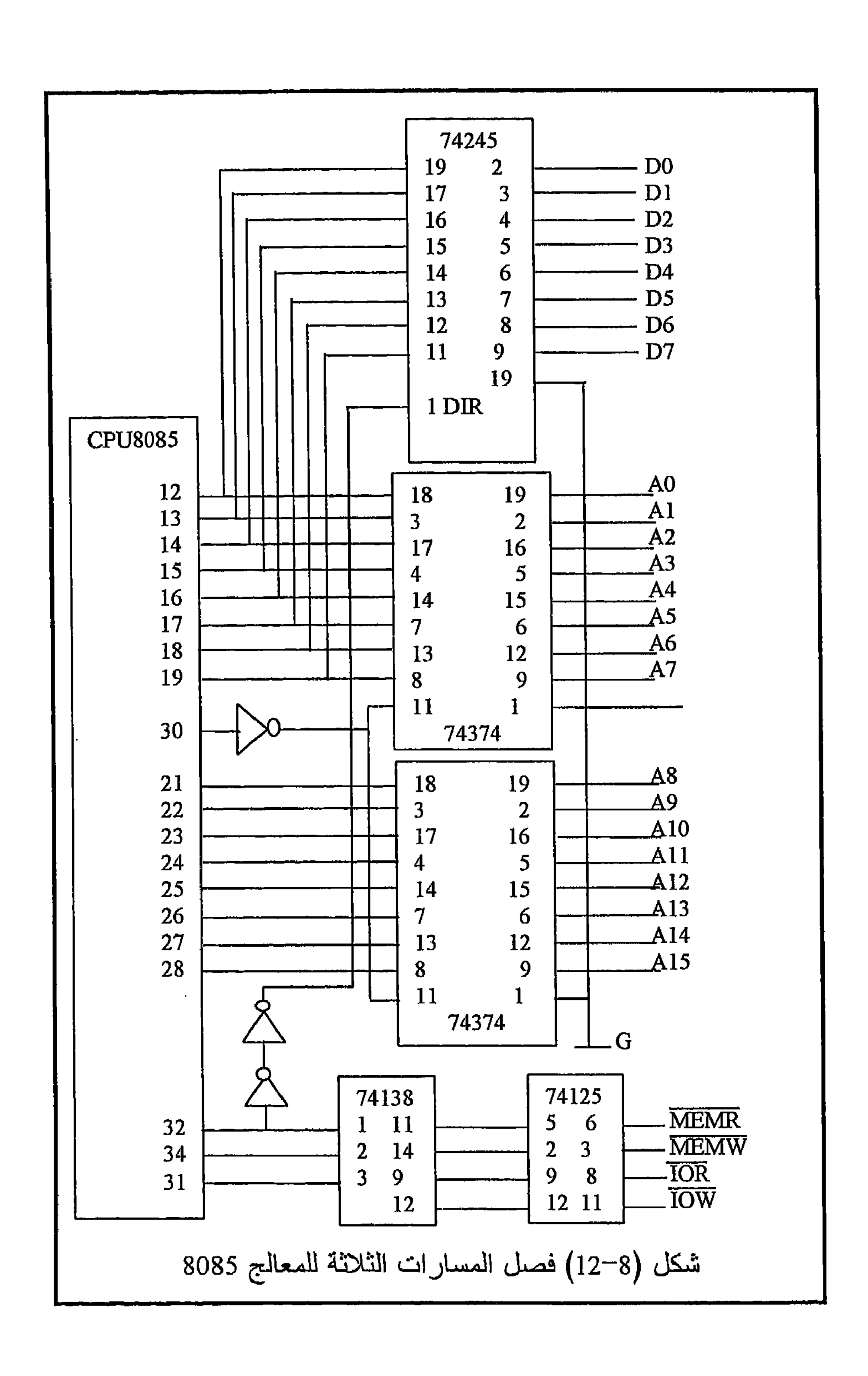
شكل (8-12) يبين المعالج 8085 وقد تم فصل buffering جميع مساراته الثلاثة وأصبحت هذه المسارات متهيئة تماما لأن يوصل عليها أى واحد من الأجهزة الخارجية مثل الذاكرة وبوابات الإدخال والإخراج كما سنرى بالتفصيل فى الفصول القادمة .



شكل (8-11) فصل خطوط التحكم الأساسية

8-5 تهيئة مسارات المعالج Z80 لعملية المواجهة

بإلقاء نظرة سريعة على أطراف المعالج Z80 كما فى شكل (8. 13) سنكتشف أن عملية المزج الزمنى time multiplexing بين مسارى العناوين والبيانات التى كانت موجودة فى المعالج 8085 غير موجودة هنا ، ولكن كل مسار متاح بصورة منفصلة عن الآخر ولذلك فإن عملية التهيئة هنا ستكون أبسط من قبل . لذلك فإن كل ما سنحتاجه هنا هو عملية فصل buffer لهذه المسارات بغرض الحماية وتوفير التيارات اللازمة للأجهزة المحيطة . شكل (8-14) يبين المعالج Z80 وقد تم فصل مساراته الثلاثة ، العناوين والبيانات والتحكم . بالنسبة لمسار العناوين فقد استخدمت الشريحة 74244 التى تتكون من ثمانى بوابات ثلاثية المنطق لهذا الغرض ، ولقد تم استخدام شريحتان منها لتحقيق عملية الفصل لل 16 خطا فى مسار العناوين . لاحظ أن خطوط التحكم فى الخرج بالنسبة لهاتين الشريحتين قد تم توصيلهما بالأرضى مباشرة مع العلم أن هذه الخطوط يمكن توصيلها على الطرف HOLD القادم من المعالج لتحقيق عملية انفصال المعالج عن المسارات الثلاثة التى سنشرحها فى موضع قادم .



شكل (8-12) فصل المسارات الثلاثة للمعالج 8085

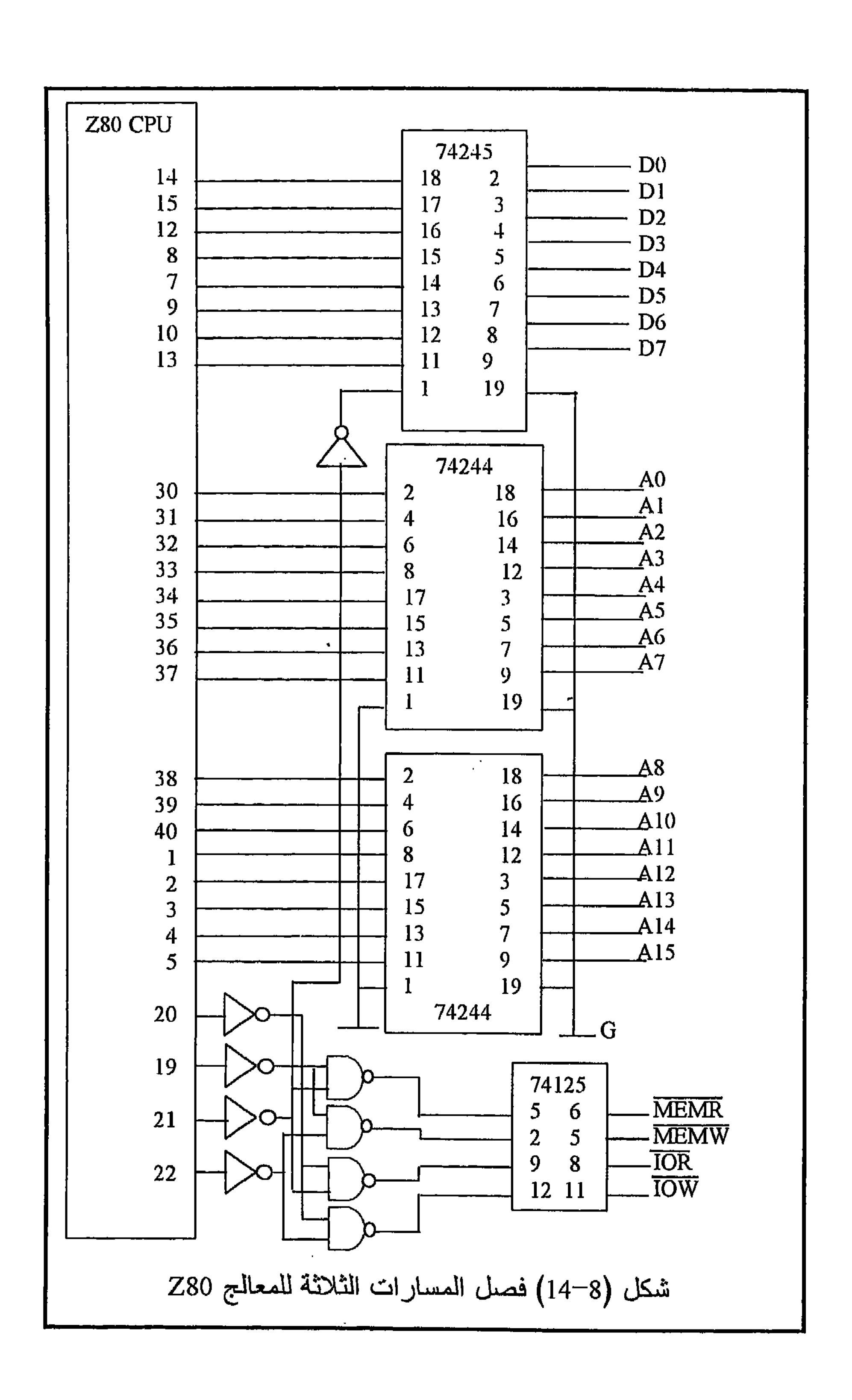
بالنسبة لمسار البيانات للمعالج Z80 فقد استخدمنا نفس الشريحة 74245 التى سبق استخدامها لعملية فصل مسار البيانات فى المعالج 8085 . هذه الشريحة كما عرفناها من قبل تتكون من ثمانى بوابات ثلاثية المنطق ثنائية الاتجاه أى تسمح بعملية فصلbuffer الإشارات التى تمر فى اتجاهين . لقد تم التحكم فى اتجاه البيانات عن طريق توصيل الطرف $\overline{DIR}$ رقم 1 فى الشريحة 74245 بالطرف $\overline{RD}$ رقم 21 فى المعالج Z80 بحيث عندما يكون الطرف $\overline{RD}$ فعال (0) فإن الشريحة 74245 ستسمح بمرور البيانات من الأجهزة المحيطة إلى المعالج . وأما عندما يكون الطرف $\overline{RD}$ غير فعال (1) فإن البيانات ستمر فى الاتجاه الآخر.

المعالج Z80

الطرف	الرقم	الرقم	الطرف
A11	1	40	A10
A12	2	39	A9
A13	3	38	A8
A14	4	37	A7
A15	5	36	A6
Clock	6	35	A5
D4	7	34	A4
D3	8	33	A3
D5	9	32	A2
D6	10	31	A1
Vcc	11	30	A0
D2	12	29	GND
D7	13	28	$\overline{RFSH}$
D0	14	27	$\overline{M1}$
D1	15	26	$\overline{RESET}$
$\overline{INT}$	16	25	$\overline{BUSREQ}$
$\overline{NMI}$	17	24	$\overline{WAIT}$
$\overline{HALT}$	18	23	$\overline{BUSAK}$
$\overline{MREQ}$	19	22	$\overline{WR}$
$\overline{IOREQ}$	20	21	$\overline{RD}$

شكل (8-13) أطراف الشريحة Z80

شكل (8-14) يبين أيضا كيفية الحصول على خطوط التحكم $\overline{MEMR}$ و $\overline{MEMW}$ و $\overline{IOR}$ و $\overline{IOW}$ باستخدام دوائر NAND . لاحظ أن الطرف $\overline{MREQ}$ يكون فعالا (0) عندما يتعامل المعالج مع الذاكرة سواء بغرض القراءة منها أو الكتابة فيها . وأما الخط $\overline{IORQ}$ فيكون فعالا (0) عندما يتعامل المعالج مع بوابات الإدخال أو الإخراج سواء بغرض القراءة أو الكتابة أيضا .



شكل (8-14) فصل المسارات الثلاثة للمعالج Z80

لاحظ أن خطوط التحكم الأربعة تكون فعالة عندما تكون صفر . بذلك نكون قد انتهينا من تهيئة جميع مسارات المعالج Z80 وأصبحت جاهزة لعملية المواجهة مع الأجهزة المحيطة .

8-6 تمارين

1. ما هو المقصود من تهيئة المسارات ؟ ولماذا تحتاج المسارات إلى تهيئة؟
2. الأجهزة المحيطة بالمعالج إما أن تكون أجهزة إدخال (ترسل ، تستقبل) المعلومات (من ، إلى) المعالج ، أو أجهزة إخراج (ترسل ، تستقبل) المعلومات (من ، إلى) المعالج . أختر كلمة مناسبة مما بين القوسين .
3. مسار البيانات ثنائى الاتجاه ، أى أن الإشارة عليه تكون خارجة من المعالج على بعض الخطوط ، وداخلة إليه على البعض الآخر (صح ، خطأ) أختر؟
4. مسار العناوين أحادى الاتجاه يحمل إشارة من الأجهزة المحيطة إلى المعالج (صح ، خطأ) أختر؟
5. الذاكرة التى يستطيع معالج من المعالجات تبلغ 64كيلوبايت لأن (مسار البيانات ، مسار العناوين) 16 بت ، أختر إجابة ؟
6. لماذا نحتاج لخطوط التحكم $\overline{MEMR}$ و $\overline{MEMW}$ ؟
7. لماذا نحتاج لخطوط التحكم $\overline{IOR}$ و $\overline{IOW}$ ؟
8. عند تنفيذ الأمر STA E100 فى المعالج 8085 أى الخطوط التالية سيكون فعالا : $\overline{MEMR}$ و $\overline{MEMW}$ و $\overline{IOR}$ و $\overline{IOW}$ ؟ أختر الخط الصحيح .
9. عند تنفيذ الأمر MOV M,A فى المعالج 8085 أى الخطوط التالية سيكون فعالا :$\overline{MEMR}$ و $\overline{MEMW}$ و $\overline{IOR}$ و $\overline{IOW}$ ؟ أختر إحدى الإجابات .
10. عند تنفيذ الأمر IN 00 فى المعالجين 8085 و Z80 أى الخطوط التالية سيكون فعالا :$\overline{MEMR}$ و $\overline{MEMW}$ و $\overline{IOR}$ و $\overline{IOW}$ ؟ أختر إحدى الإجابات .
11. عند تنفيذ الأمر OUT 00 فى المعالجين 8085 و Z80 أى الخطوط التالية سيكون فعالا :$\overline{MEMR}$ و $\overline{MEMW}$ و $\overline{IOR}$ و $\overline{IOW}$ ؟ أختر إحدى الإجابات .
12. .

```
xxx:    MVI A,89H
        STA E100
        IN 00
        OUT 00
        LXI H, E100
        MOV B,M
        JMP xxx
```

8085 Program

```
xxx:    LD A,89H
        LD (E100), A
        IN 00
        OUT 00
        LD HL, E100
        MOV B,M
        JP xxx
```

Z80 Program

البرنامج السابق عبارة عن حلقة لا نهائية ، ارسم شكل الإشارة مع الزمن علـــى كل خط من الخطوط $\overline{MEMR}$ و $\overline{MEMW}$ و $\overline{IOR}$ و $\overline{IOW}$ فى أثناء تنفيذ هـــذا البرنامج ؟

13. لو نفذنا برنامج السؤال 12 مرة واحدة ، أختر الإجابة الصحيحة فى كل ممـــا يلى :

عدد نبضات الفعالية للخط $\overline{MEMR}$ سيكون (17 ، 13 ، 19) .

عدد نبضات الفعالية للخط $\overline{MEMW}$ سيكون (17 ، 1 ، 9) .

عدد نبضات الفعالية للخط $\overline{IOW}$ سيكون (17 ، 1 ، 9) .

عدد نبضات الفعالية للخط $\overline{IOR}$ سيكون (17 ، 1 ، 9) .

14. إذا كان الخط $IO/\overline{M}=1$ والخط $\overline{WR}=0$ فإن ذلك يعنى (قراءة ، كتابة) (فـى ، من) (ذاكرة ، بوابة إدخال ، بوابة إخراج) أختر الإجابة الصحيحة ؟

15. إذا كان الخط $IO/\overline{M}=1$ والخط $\overline{RD}=0$ فإن ذلك يعنى (قراءة ، كتابة) (فـى ، من) (ذاكرة ، بوابة إدخال ، بوابة إخراج) أختر الإجابة الصحيحة ؟

16. ما المقصود بالمشاركة الزمنية بين خطوط البيانات وخطوط العنـــاوين فـــى المعالج 8085 ؟ وما الهدف منها ؟

17. هل هذه المشاركة موجودة فى المعالج Z80 ؟

18. اشرح دور الخط ALE فى عملية فصل إشارة العناوين عن إشارة البيانـــات على الخطوط AD0-AD7 فى المعالج 8085 ؟

19. اشرح كيفية الحصول على خطوط التحكم الأربعة فى المعالج Z80 ؟

الفصل التاسع

مواجهة الذاكرة

Memory Interfacing

9-1 مقدمة

إن ذاكرة الكومبيوتر تكون عادة ذاكرة إليكترونية أى أن طريقة مسك المعلومة وحفظها يتم إليكترونيا ، وذلك على العكس من الأنواع الأخرى من الذاكرة مثل شرائط الكاسيت والأقراص الممغنطة والتى يتم مسك المعلومة عليها مغناطيسيا ، ونحن فى هذا الفصل لن نتعرض للتركيب الإلكترونى للذاكرة ولكن كل ما يهمنا هنا هو معرفة كيفية توصيل أو مواجه المعالج مع شرائح الذاكرة .

إن دليل التليفونات يمكن النظر إليه على أنه نوع من أنواع الذاكرة التى تسجل فيها المعلومات فى صورة حروف هجائية ، وهذا النوع يماثل فى الخواص ذاكرة القراءة فقط ROM والتى تستخدم فى الحاسبات ، وحيث أنه يمكن قراءة المعلومات من الدليل ولكن لا يمكن تغيرها فكذلك يمكن قراءة المعلومات من ال ROM ولا يمكن الكتابة فيها ولذلك سميت بذاكرة القراءة فقط Read Only Memory . ذلك على العكس من شريط الكاسيت أو القرص الممغنط حيث يمكن تخزين المعلومات عليهما كما يمكن مسحها أو تغيرها فى أى لحظة مثلها فى ذلك مثل ذاكرة القراءة والكتابة فى الحاسبات والتى سميت عرفا بذاكرة الاتصال العشوائى Random Access Memory; RAM . أى نظام من نظم الحاسبات لابد وأن يحتوى على كل من النوعين من أنواع الذاكرة (ROM و RAM) حيث تحتوى ال ROM على الثوابت والبرامج المهمة لتشغيل نظام الحاسب والتى يمنع المستخدم من الدخول عليها نظرا لخطورة التغيير أو التعديل فيها ، وأما ال RAM فهى الذاكرة التى تكون متاحة للمستخدم ليقوم فيها بتخزين جميع بياناته أو برامجه فى حالة تعامله مع النظام . لاحظ أن ال ROM من أهم خواصها أنه عند انقطاع القدرة (الكهرباء) أى عند إطفاء النظام فإن جميع ما بها من معلومات لا تفقد ولكن تظل محفوظة ، وذلك على العكس من ال RAM التى تفقد كل محتوياتها بمجرد إطفاء النظام ، ولذلك فإنه قبل إطفاء النظام لابد من تخزين محتويات ال RAM التى نحتاجها على ذاكرة مستديمة مثل الشرائط أو الأقراص الممغنطة . شكل (9-1) يبين مناظرة بين ذاكرة الحاسبات والذاكرة بمعناها العام .

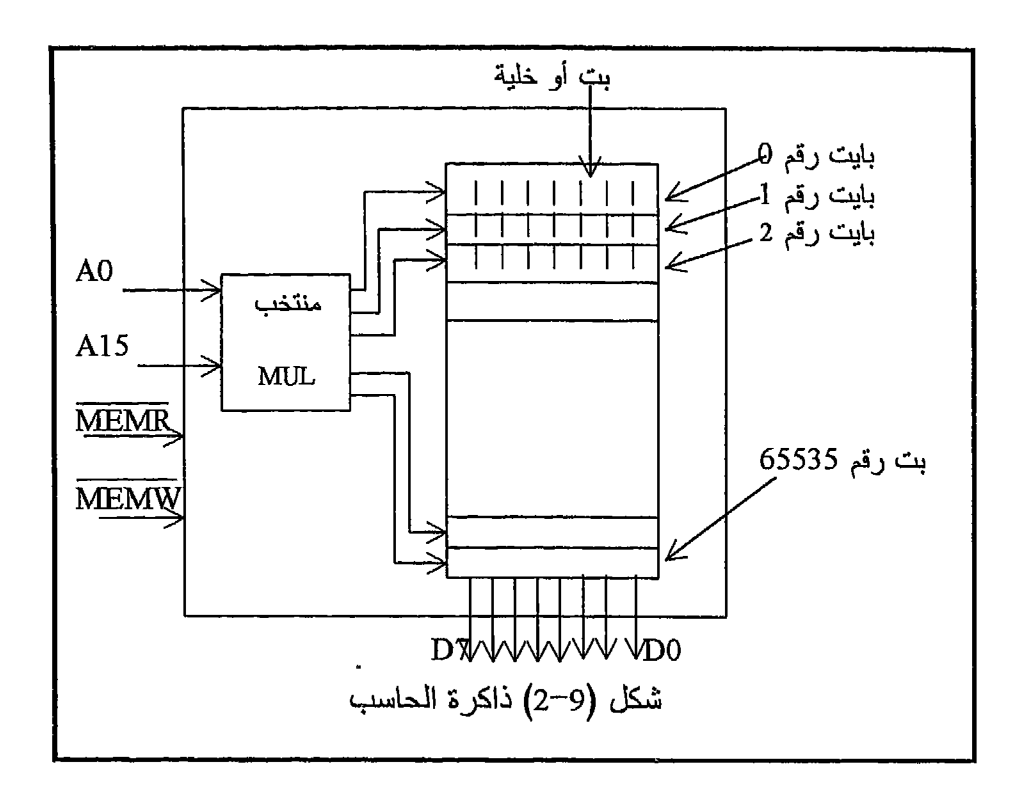
إن كل ذاكرة مهما كانت لابد وأن هناك طريقة معينة تسهل عملية استدعاء المعلومات منها ، فمثلا من دليل التليفونات يمكنك الوصول إلى رقم تليفون أى شخص عن طريق التسلسل الأبجدى للأحرف ، أما على شريط الكاسيت فيمكنك تحديد مكان المعلومة عن طريق عداد الشريط ، وأما فى ذاكرة الحاسب فإن كل مكان من أماكن الذاكرة محدد بعنوان وهذا العنوان عبارة عن شفرة من الوحايد والأصفار توضع على مسار العناوين فتسبب عملية تنشيط أو إثارة لمكان واحد فقط من أماكن الذاكرة ليصبح جاهزا للتعامل معه بواسطة المعالج .

الذاكرة	
ذاكرة الكومبيوتر	الذاكرة عامة
ROM تحتوى البرامج والثوابت التى يمنع المستخدم من الوصول إليها أو العبث بها نظرا لأهميتها لتشغيل النظام ويمكنه فقط أن يقرأها.	دليل التليفونات والاسطوانات Pick up كلها أمثلة على الذاكرة التى يمنع الكتابة فيها ولكن يسمح فقط بقراءتها.
RAM يمكن للمستخدم أن يقرأ محتوياتها ويسجل فيها ما يشاء ، ويمسح ويضيف فى أى مكان فيها .	شريط الكاسيت وشريط الفيديو يمكن التسجيل فيهما وقراءة محتوياتهما ، كما يمكن المسح أو الإضافة لأى جزء فيهما .
يتم بناء هذه الذاكرة عادة من أشباه الموصلات ، وهى فى العادة عبارة عن قلابات flip flops .	يتم بناؤها من أوراق مثل دليل التليفون ، أو شرائط من مواد مغناطيسية مثل شرائط الكاسيت والفيديو .

شكل (9-1) مناظرة بين ذاكرة الحاسب والذاكرة عامة

9-2 أساسيات بناء ذاكرة الحاسب

شكل (9-2) يبين أساسيات بناء الذاكرة . يتكون مسار العناوين القادم من أى واحد من المعالجات التى ندرسها الآن من 16 خطا أو 16 بت ، ولذلك فإن كمية الذاكرة التى يستطيع المعالج أن يتعامل معها تقدر ب 2^{16} أى 65536 بايت مرتبة كما لو كانت أرفف فى دولاب وكل رف من هذه الأرفف يمثل بايت وكل بايت أو رف تتكون من ثمانى بتات أو خلايا كما فى شكل (9-2) . كل واحدة من هذه البايتات تعرف بعنوان خاص بها ولذلك فإن أول بايت عنوانها هو (صفر) وآخر بايت أو آخر رف فى هذا الدولاب عنوانه هو 65535 وذلك فى النظام العشرى . عند التعامل مع أى بايت من هذه البايتات أى القراءة منها أو الكتابة فيها فإن ذلك يكون على البايت الكاملة وليست هناك وسيلة للتعامل مع جزء من البايت ، أى عدد معين من بتاتها دون الباقى .



شكل (9-2) ذاكرة الحاسب

لاحظ أن أى شفرة على خطوط العناوين A0 إلى A15 ستحدد عنوان مكان أو بايت من بايتات الذاكرة فى النظام الثنائى ، لاحظ أيضا أنه طالما أن كل بايت تتكون من ثمانى بتات فإنها تتوافق مع مسار البيانات الذى هو ثمانى بتات أو ثمانية خطوط أيضا والذى سيحمل المعلومات من أو إلى هذه البايتات . بعد أن يضع المعالج عنوان البايت التى سيتعامل معها على مسار العناوين فإنه لابد وأن يحدد طريقة التعامل مع هذه البايت إذا كانت قراءة أو كتابة فيها. إذا كان المعالج يريد الكتابة فى الذاكرة فإنه يجعل خط التحكم $\overline{MEMW}$ (Memory write) وهو خط الإعلان عن الكتابة فى الذاكرة فعالا ، أما إذا كان يريد القراءة منها فإنه يجعل خط التحكم $\overline{MEMR}$ (Memory read) وهو خط الإعلان عن القراءة من الذاكرة فعالا ، لذلك فإنه لزم إضافة هذين الخطين ، $\overline{MEMW}$ و $\overline{MEMR}$ فى شكل (9-2) . تذكر دائما أن معنى وضع خط فوق اسم أى إشارة يعنى أن هذه الإشارة تكون فعالة حينما تكون صفرا Low وهى الحالة الموجودة فى الإشارتين $\overline{MEMW}$ و $\overline{MEMR}$. شكل (9-3) يبين بعض العناوين وشفراتها الثنائية الستعشرية . حاول دراسة هذا الجدول وأضف من عندك شفرات لبعض العناوين الغير مذكورة فى الجدول . لاحظ أنه فى أثناء البرمجة وفى كل تعاملاتنا مع

العناوين فيما بعد سيكون فى الصورة الستعشرية التى تتكون من أربع خانات كما هو مبين فى شكل (9-3).

الشفرة الموجودة على خطوط العناوين A0 الى A15				نظام ستعشرى	نظام عشرى
15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0		
0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0000	0
0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 1	0001	1
0 0 0 0	0 0 0 0	0 0 0 0	0 0 1 0	0010	2
0 0 0 0	0 0 0 0	0 0 0 0	0 0 1 1	0011	3
0 0 0 0	0 0 0 0	0 0 0 0	0 1 0 0	0100	4
0 0 0 0	0 0 0 0	0 0 0 0	0 1 0 1	0101	5
....	.....	.....	......	....	....
......	......	....	....	....	....
0 1 0 1	1 1 0 1	1 1 1 1	1 0 1 1	5DFB	24059
......	....	....	....	....	......
1 0 1 0	1 1 0 1	1 0 1 0	1 1 0 0	ADAC	44460
....	......	......	....	....	....
....	....	....	....	....	......
1 1 1 1	1 1 1 1	1 1 1 1	1 1 1 0	FFFE	65534
1 1 1 1	1 1 1 1	1 1 1 1	1 1 1 1	FFFF	65535

شكل (9-3) عنونة الذاكرة المواجهة للمعالج

شكل (9-2) يبين أيضا كيفية الاتصال بمكان معين فى الذاكرة عن طريق منتخب البايتات الموجود داخل شريحة الذاكرة . يقوم هذا المنتخب Address decoder باختيار أو انتخاب واحد من الخطوط الموجودة على خرجه حيث يتم هذا الاختيار على أساس الشفرة الموجودة على دخله من مسار العناوين وجعله فعالا وهذا الخط بالتالى يختار البايت المقابلة له وإما يخرج محتوياتها على مسار البيانات إذا كان الخط $\overline{MEMW}$ فعالا أو يدخل محتويات مسار البيانات إلى هذه البايت إذا كان الخط $\overline{MEMR}$ فعالا .

إن الزمن المأخوذ لوضع محتويات أى بايت من بايتات الذاكرة على مسار البيانات أو العكس يسمى زمن الاتصال بالذاكرة Memory access time وهذا الزمن يعتبر خاصية من خواص شريحة الذاكرة حيث يختلف فى طوله وقصره على حسب التكنولوجيا والمادة المستخدمة فى تصنيع الشريحة ، وعادة يكون هذا الزمن فى حدود المائة نانو ثانية (nano second) حيث النانو ثانية تساوى 10^{-9} من الثانية .

إن وحدات قياس سعة الذاكرة فى عالم الحاسبات هو الكيلو بايت ك.ب KB . وتم التعارف على أن الواحد كيلوبايت يساوى 1024 بايت بدلا من 1000 التى

تستخدم دائما مع تعريف الكيلو فى الحياة العملية وذلك لسهولة التعامل مع الرقم 1024 فى النظام الثنائى والستعشرى . كما نعلم فإن شرائح المعالجات التى نتعامل معها لها 16 خطا للعناوين وبهذا العدد من خطوط العناوين فإنه يمكن لهذه الشريحة التعامل مع 65536 مكان من أماكن الذاكرة كما رأينا منذ قليل . لاحظ أن 65536 عند قسمتها على 1024 تعطى 64 كيلوبايت لذلك يقال دائما إن هذه المعالجات يمكنها التعامل مع 64 كيلوبايت ذاكرة . شكل (9-4) يبين علاقة عدد خطوط العناوين بكمية الذاكرة التى يمكن لأى معالج أن يتعامل معها وكيف أن كمية هذه الذاكرة تتضاعف مع كل زيادة فى عدد خطوط العناوين بمقدار خط واحد .

كمية الذاكرة (بايت)	عدد خطوط العنونة المطلوبة
2	1
4	2
8	3
16	4
32	5
64	6
128	7
256	8
512	9
1024 (1 كيلوبايت)	10
2048 (2 كيلوبايت)	11
4096 (4 كيلوبايت)	12
8192 (8 كيلوبايت)	13
16384 (16 كيلوبايت)	14
32768 (32 كيلوبايت)	15
65536 (64 كيلوبايت)	16

شكل (9-4) مضاعفة كمية الذاكرة بزيادة خطوط العناوين بمقدار خط واحد

9-3 كيف سنوصل الذاكرة على المعالج؟

يوجد فى الأسواق العديد من شرائح الذاكرة التى تختلف من شريحة لأخرى من حيث كمية الذاكرة الموجودة فى كل شريحة . فهناك شرائح تحتوى الواحدة منها على واحد كيلوبايت وأخرى تحتوى الواحدة منها على 512 بايت وأخرى تحتوى الواحدة منها على 4 كيلوبايت وهكذا ، بل إن هناك شرائح تحتوى الواحدة منها على64 كيلو بايت وأكثر ، السؤال الآن أى هذه الشرائح نستخدم للحصول على ال 64 كيلو بايت التى سنوصلها على المعالج ؟ وما هى أفضل الطرق لتوصيل هذه الشرائح ؟

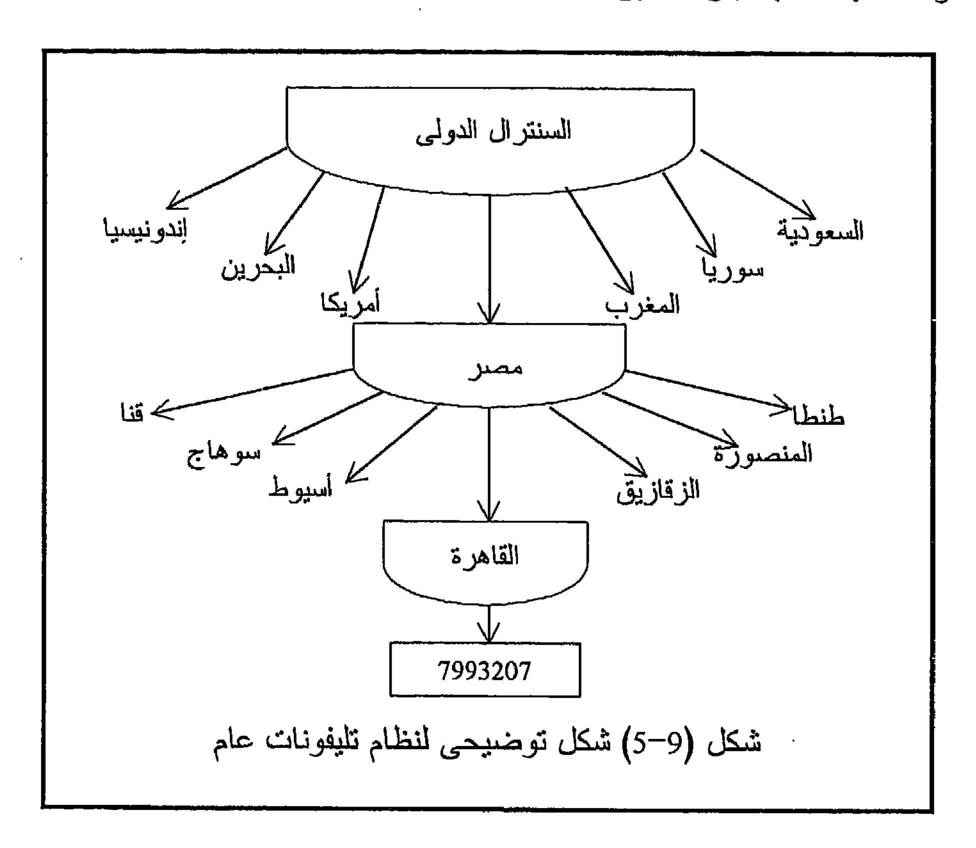
9-3-1 مثال توضيحى

افترض أننا فى المملكة العربية السعودية ونريد الاتصال هاتفيا بالرقم 7993207 الموجود في مصر بمدينة القاهرة . إننا لكى نفعل ذلك لابد أن نضرب أولا رقم مصر فى السنترال الدولى وهو 002 ثم نضرب رقم مدينة القاهرة فى مصر وهو 02 ثم نضرب الرقم الذى معنا ، أى أن الرقم كله الآن أصبح كالتالى 002027993207 . على ضوء ذلك فإننا يمكننا النظر إلى نظام التليفونات فى العالم على النحو المبين فى شكل (9-5) حيث يتكون هذا النظام من سنترال عالمى يحتوى رقما لكل دولة من دول العالم وبمجرد ضرب رقم أى واحدة من هذه الدول فإنه يوصلك بسنترال عموم هذه الدولة الذى يحتوى رقم لكل مدينة داخل هذه الدولة ، وبمجرد ضرب رقم أى مدينة من هذه المدن فإنه يوصلك بسنترال هذه المدينة الذى يحتوى جميع الأرقام داخل هذه المدينة ومنها الرقم الذى تريده . السؤال الآن لماذا هذا التعب فى شرح نظام التليفونات وما دخله بموضوع توصيل شرائح الذاكرة على المعالج ؟ إن الشبه كبير جدا بين الاثنين فكما أنك تستطيع النظر لأى رقم تليفون وتقوم بتقسيمه إلى عدة أجزاء حيث جزء منه يمثل الرقم الدولى وجزء يمثل رقم المدينة داخل الدولة وجزء يمثل رقم التليفون داخل المدينة فكذلك يمكن عمل نفس الشيء مع أى عنوان من عناوين الذاكرة كما سنرى بعد قليل .

9-3-2 نظام بلوكات الذاكرة

أول ما سنفعله لبناء ذاكرة مقدارها 64 كيلوبايت هو تقسيم هذه الكمية من الذاكرة إلى عدد من البلوكات يحتوى كل بلوك منها على عدد من الكيلوبايتات . فمثلا يمكننا تقسيمها إلى 64 بلوك يحتوى كل منها على واحد كيلوبايت ، أو إلى 8 بلوكات يحتوى الواحد منها على 8 كيلوبايت ، أو إلى 128 بلوك يحتوى الواحد منها على نصف كيلوبايت وهكذا فإن عدد البلوكات سيترك تماما للمستخدم

الحرية فى تحديده . بمجرد تحديد عدد البلوكات سيتحدد فورا كـم خطـا مـن خطوط مسار العناوين سيستخدم لتمييز البلوكات بعضها من بعض وكـم خطـا سيستخدم لتمييز البايتات داخل كل بلوك بحيث ستدخل خطوط عنونة أو تمييـز البلوكات على منتخب يكون خرجه 2 أس عدد هذه الخطوط بحيث سيذهب كـل واحد منها لتنشيط بلوك معين .



شكل (9-5) شكل توضيحى لنظام تليفونات عام

شكل (9-6) يبين تقسيم خطوط مسار العناوين إلـى خطـوط عنونـة بلوكـات وخطوط عنونة بايتات داخل البلوكات فى حالتين ، الأولى فى حالة تقسيم الذاكرة إلى 64 بلوك يحتوى الواحد منها على واحد كيلوبايت ، والثانية فى حالة تقسـيم الذاكرة إلى 8 بلوكات يحتوى الواحد منها على 8 كيلوبايت . فـى المثـال الأول طالما أن وحدة البناء وهى البلوك تساوى واحد كيلوبايت لذلـك فإنـه يلـزم 10 خطوط (2^{10}= 1024 = واحد كيلوبايت) من ال 16 خطا الموجودة فـى مسـار العناوين لعنونة ال 1024 بايت الموجودة داخل البلوك . أما ال 6 خطوط الباقيـة من مسار العناوين (2^{6}= 64) فإنها تستخدم فى عنونة أو اختيار واحد مـن ال 64 بلوك . المثال الثانى فى شكل (9-6) يبين عنونـة ذاكـرة مـن 64 كيلوبـايت باستخدام بلوكات كل منها يحتوى على 8 كيلوبايت . لاحظ أن عـدد البلوكـات

يساوى 64 مقسوما على عدد الكيلوبايتات فى البلوك الواحد . إن خطوط العنونة داخل البلوك ستكون 13 خطا فى هذه الحالة حيث سيبقى ثلاثة خطوط من ال 16 خطا لعنونة الثمانية بلوكات . خطوط عنونة البلوكات ستتصل بمنتخب يقوم باختيار واحد من البلوكات الثمانية على حسب الشفرة الموجودة على هذه الخطوط وتسمى الخطوط الخارجة من هذا المنتخب بخطوط اختيار البلوك أو خطوط تنشيط البلوك . شكل (9-7) يبين رسما تخطيطيا لبناء ذاكرة من 64 كيلوبايت باستخدام بلوكات 8 كيلوبايت .

خطوط مسار العناوين

A15 A14 A13 A12 A11 A10	A9 A8 A7 A6 A5 A4 A3 A2 A1 A0
ستة خطوط لعنونة أو اختيار البلوك	عشرة خطوط للعنونة داخل البلوك

أ- عنونة ذاكرة 64 ك.ب مكونة من 64 بلوكات كل منها 1 ك.ب

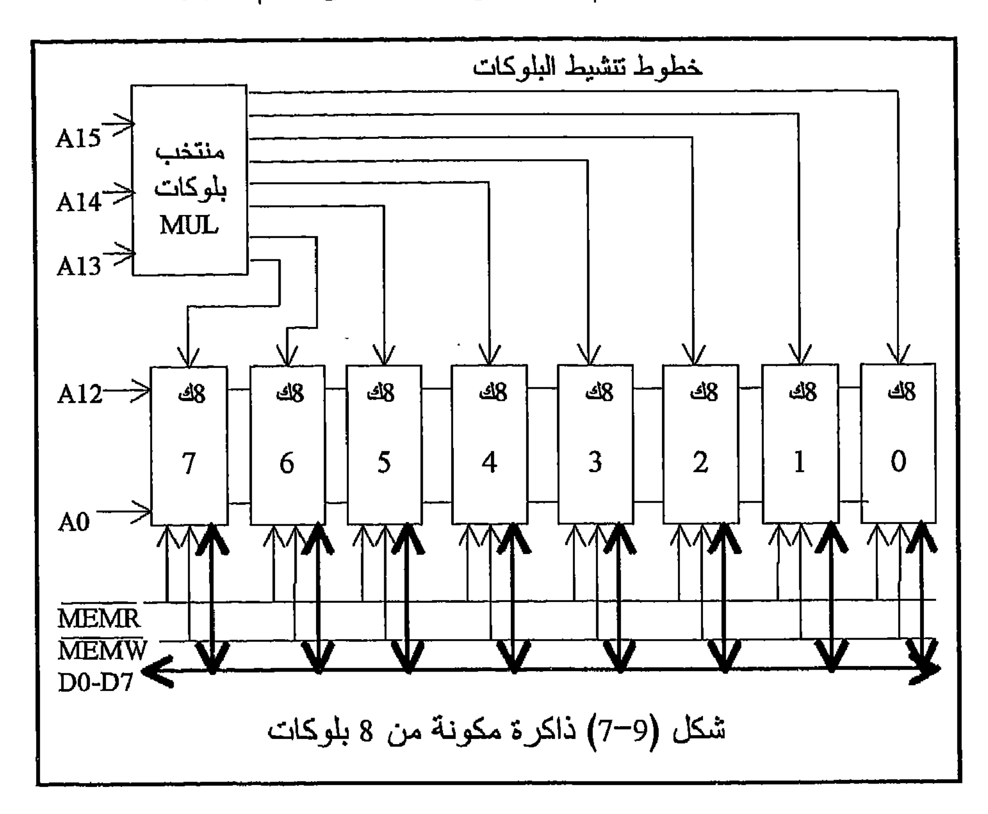
A15 A14 A13	A12 A11 A10 A9 A8 A7 A6 A5 A4 A3 A2 A1 A0
3 خطوط لعنونة 8 بلوكات	13 خطا للعنونة داخل كل بلوك

ب-عنونة ذاكرة 64 ك.ب مكونة من 8 بلوكات كل منها 8 ك.ب

شكل (9-6) تقسيم الذاكرة إلى بلوكات ثم عنونتها

من شكل (9-7) نلاحظ أن وظيفة منتخب البلوكات هى اختيار واحد من البلوكات على حسب الشفرة الموجودة على خطوط العناوين A13 و A14 و A15 وتنشيطه أى جعله جاهزا لاستقبال أو إرسال معلومات . لاحظ أيضا أن جميع البلوكات متصلة بخطوط العناوين A0 إلى A12 لتحديد أى بايت داخل البلوك الذى تم اختياره سيتم التعامل معها . لاحظ أيضا اتصال كل بلوك بخطوط التحكم $\overline{MEMR}$ و $\overline{MEMW}$ ومسار البيانات D0 إلى D7 . مقارنة سريعة بين نظام تقسيم التليفونات العالمى فى شكل (9-5) ونظام تقسيم الذاكرة إلى بلوكات فى شكل (9-7) نرى أن منتخب البلوكات فى شكل (9-7) يلعب نفس الدور الذى يلعبه السنترال الدولى والذى يقوم بتوصيلك إلى دولة معينة على حسب الرقم الداخل له ، فكذلك منتخب البلوكات على حسب الرقم أو الشفرة الموجودة على

الخطوط A13 إلى A15 يقوم بتوصيل المعالج على واحد من هذه البلوكات . بعد قليل سنرى كيفية تقسيم البلوكات إلى شرائح كل منها يحتوى عددا معينا من البايتات والتى تناظر عملية تقسيم الدول إلى مدن كما فى نظام التليفونات .



شكل (9-7) ذاكرة مكونة من 8 بلوكات

بمجرد تحديد عدد البلوكات من قبل المستخدم فإن المدى العنوانى لكل بلوك يتحدد على حسب هذا العدد . المدى العنوانى لأى بلوك يتحدد بتحديد عنوان البداية وعنوان النهاية لكل بلوك ، لاحظ أن عنوان البداية لأى بلوك يبدأ من حيث انتهى البلوك السابق له . شكل (9-8) يبين المدى العنوانى للثمانية بلوكات الموجودة فى شكل (9-7) وهو المثال الذى سنفترضه دائما فى عملية بناء الذاكرة .

9-3-3 بناء البلوكات من شرائح

نحن نعلم أن الذاكرة تكون فى هيئة شرائح وكل شريحة لها سعة معينة ، فهناك مثلا شرائح سعتها 1 كيلو بايت وأخرى سعتها 2 كيلوبايت أو 512 بايت وأحيانا تجد 64 كيلوبايت على شريحة واحدة ، فما علاقة هذه الشرائح بالبلوكات أو وحدات البناء التى تكلمنا عنها سلفا؟ لما كان البلوك وحدة البناء فى هيكل الذاكرة

الكلى فإن الشريحة ستكون وحدة البناء داخل البلوك ، أى أن البلوك يتكون من عدد من الشرائح وهذا العدد سيتوقف على سعة الشرائح المستخدمة ومن المفضل أن تكون جميع الشرائح داخل البلوك الواحد لها نفس السعة . فمثلا البلوك ذى الثمانية كيلوبايت الذى استخدمناه سلفا يمكن بناؤه من 4 شرائح كل منها 2 كيلو أو من 8 شرائح كل منها 1 كيلو أو من شريحة واحدة سعتها 8 كيلوبايت وهكذا .

رقم البلوك	خطوط العناوين A0 إلى A15																العنوان ستعشرى
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	**0**	**0**	**0**	0	0	0	0	0	0	0	0	0	0	0	0	0	0000
	0	**0**	**0**	1	1	1	1	1	1	1	1	1	1	1	1	1	1FFF
1	**0**	**0**	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	2000
	0	**0**	**1**	1	1	1	1	1	1	1	1	1	1	1	1	1	3FFF
2	**0**	**1**	**0**	0	0	0	0	0	0	0	0	0	0	0	0	0	4000
	0	**1**	**0**	1	1	1	1	1	1	1	1	1	1	1	1	1	5FFF
3	**0**	**1**	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	6000
	0	**1**	**1**	1	1	1	1	1	1	1	1	1	1	1	1	1	7FFF
4	**1**	**0**	**0**	0	0	0	0	0	0	0	0	0	0	0	0	0	8000
	1	**0**	**0**	1	1	1	1	1	1	1	1	1	1	1	1	1	9FFF
5	**1**	**0**	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	A000
	1	**0**	**1**	1	1	1	1	1	1	1	1	1	1	1	1	1	BFFF
6	**1**	**1**	**0**	0	0	0	0	0	0	0	0	0	0	0	0	0	C000
	1	**1**	**0**	1	1	1	1	1	1	1	1	1	1	1	1	1	DFFF
7	**1**	**1**	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	E000
	1	**1**	**1**	1	1	1	1	1	1	1	1	1	1	1	1	1	FFFF

شكل (9-8) المدى العنوانى لكل بلوك من الثمانية بلوكات فى شكل (9-7)

السؤال الآن هو: كيف يتم توصيل هذه الشرائح داخل البلوك الواحد؟ إن طريقة توصيل الشرائح داخل البلوك هى نفسها طريقة توصيل البلوكات للحصول على هيكل الذاكرة . أى أن خطوط العناوين الداخلة للبلوك جزء منها سيستعمل لعنونة البايتات المختلفة داخل الشريحة والجزء الآخر سيستعمل لاختيار أو انتخاب الشرائح . إن عدد الخطوط فى كل جزء من هذه الأجزاء سيتحدد على حسب سعة الشرائح المستخدمة ولنفرض البلوك ال 8 كيلوبايت كمثال . إذا استخدمنا شرائح سعة كل منها 2 كيلوبايت فى بناء هذا البلوك فإن كل شريحة ستحتاج إلى 11 خطا (A0 إلى A10) من ال 13 خطا الداخلة إلى البلوك ويتبقى خطان (A11

و A12) يستخدمان فى عملية اختيار الشرائح عن طريق منتخب آخر نسميه منتخب اختيار الشرائح . لاحظ أنه باستخدام خطين يمكن اختيار واحدة من أربع شرائح حيث 2^2 يساوى 4 . شكل (9-9) عبارة عن مثال لبيان المدى العنوانى لكل شريحة من شرائح البلوك الأول ، وحاول أنت كتابة المدى العنوانى لشرائح واحد من البلوكات الأخرى . من شكل (9-9) نلاحظ الآتى :

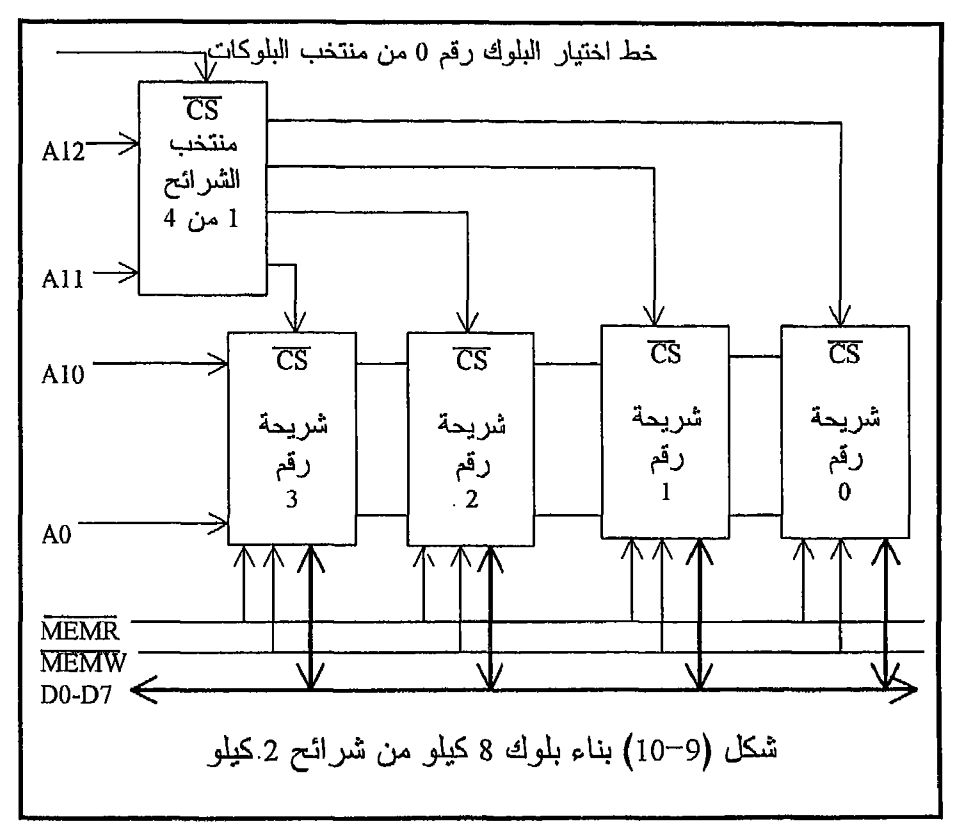
- أول عنوان فى أول شريحة فى البلوك هو أول عنوان فى البلوك ، فمثلا فى شكل (9-9) العنوان الأول فى أول شريحة فى أول بلوك هو 0000 .
- آخر عنوان فى آخر شريحة فى البلوك هو آخر عنوان فى البلوك ، فمثلا فى شكل (9-9) آخر عنوان فى آخر شريحة فى البلوك الأول هو 1FFF وهو آخر عنوان فى أول بلوك كما فى شكل (9-8) .
- خطوط اختيار البلوك A13 وA14 و A15 كانت أصفارا دائما لأننا فى البلوك الأول الذى مداه العنوانى 0000 إلى 1FFF .
- خطوط اختيار الشريحة ثابتة لا تتغير طول مدى الشريحة ، فالشريحة الأولى خطوط اختيارها كانت 00 والشريحة الثانية خطوط اختيارها كانت 01 وهكذا .
- أول عنوان فى الشريحة نحصل عليه بأن نضع جميع خطوط اختيار البايت داخل الشريحة (A0 إلى A10) تساوى أصفارا ، وآخر عنوان فى الشريحة نحصل عليه بأن نضع هذه الخطوط تساوى وحايد .

خطوط العناوين A0 إلى A15																المدى	رقم
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	العنوانى	الشريحة
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000	0
0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	07FF	
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0800	1
0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0FFF	
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1000	2
0	0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	17FF	
0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1800	3
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1FFF	

شكل (9-9) المدى العنوانى لكل شريحة داخل البلوك الأول

شكل (9-10) يبين التركيب الداخلى لأحد البلوكات الثمانية المبينة فى شكل (9-7) حيث يتكون هذا البلوك من أربع شرائح كل منها 2 كيلوبايت ولذلك فإن كل شريحة من هذه الشرائح لها11 خطا للعناوين متصلة بالخطوط A0 إلى A10 القادمة من مسار عناوين المعالج . كذلك فإن كل شريحة كما نرى لابد وأن تكون متصلة بمسار العناوين D0 إلى D7 . إن منتخب الشرائح الموجود فى

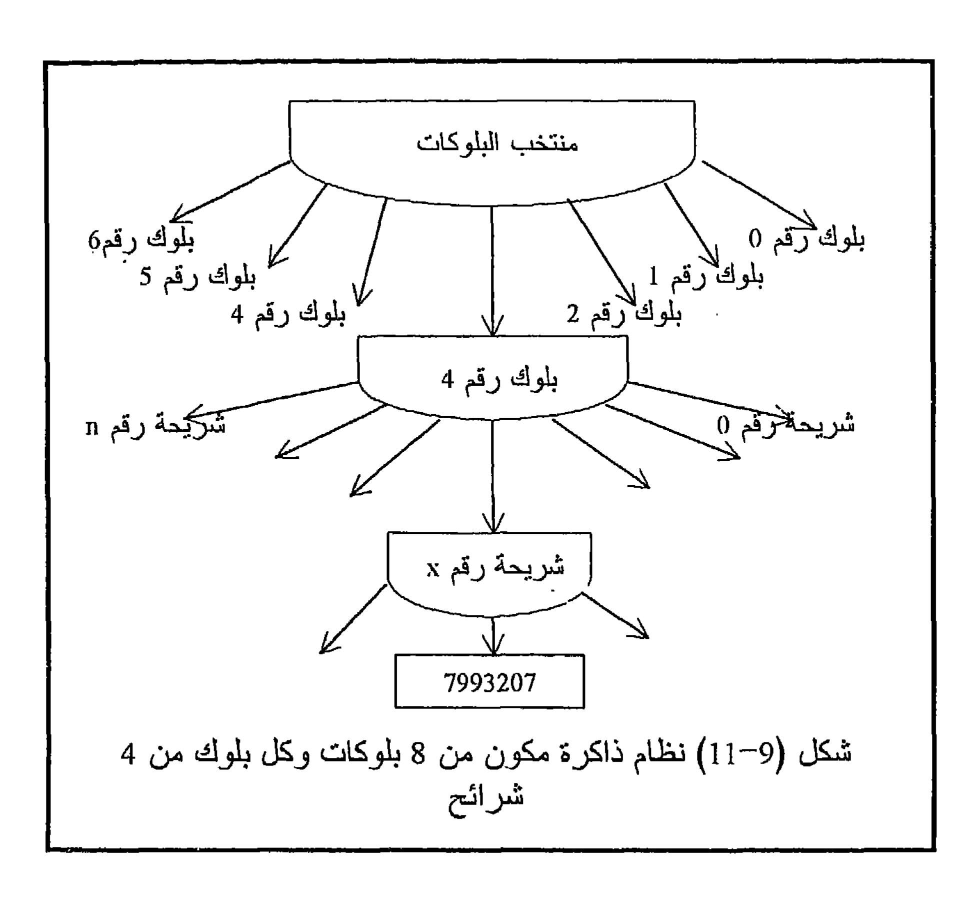
شكل (9-10) وبناء على الشفرة الموجودة على دخليه A11 و A12 سيكون واحد فقط من خروجه فعالا وباقى الخروج خاملة مما سيجعل شريحة الذاكرة المتصلة بهذا الخرج هى فقط الفعالة وأما باقى الشرائح فستكون خاملة ، وعلى ذلك فـإن هذه الشريحة وبناء على الشفرة الموجودة على الخطوط A0 إلى A10 سـتكون إحدى بايتاتها جاهزة للتعامل مع المعالج إما للكتابة أو للقراءة على حسب حالـة الخطين $\overline{MEMR}$ ، و $\overline{MEMW}$. لاحظ أن منتخب الشرائح فى شكل (9-10) لن يكون فعالا إلا إذا جاءت له إشارة أو نبضة من منتخب البلوكات فى شكل (9-7) تخبره أن هذا البلوك قد اختير للتعامل مع المعالج وبذلك يصبح منتخب الشـرائح فعالا .



شكل (9-10) بناء بلوك 8 كيلو من شرائح 2 كيلو

شكل (9-11) يبين نظام ذاكرة مكون من 8 بلوكات وكل بلـوك مكـون مـن 4 شرائح وكل شريحة مكونة من 2 كيلوبايت فهل نستطيع الآن مقارنة هذا الشـكل بشكل (9-5) الذى يبين رسما توضيحيا لنظام تليفونـات عـالمى؟ إن منتخـب البلوكات يناظر السنترال الدولى ومنتخب الشرائح يناظر سنترال الدولة التى تـم اختيارها وعنوان البايت داخل الشريحة يناظر رقم أى تليفون داخل الدولة التـى

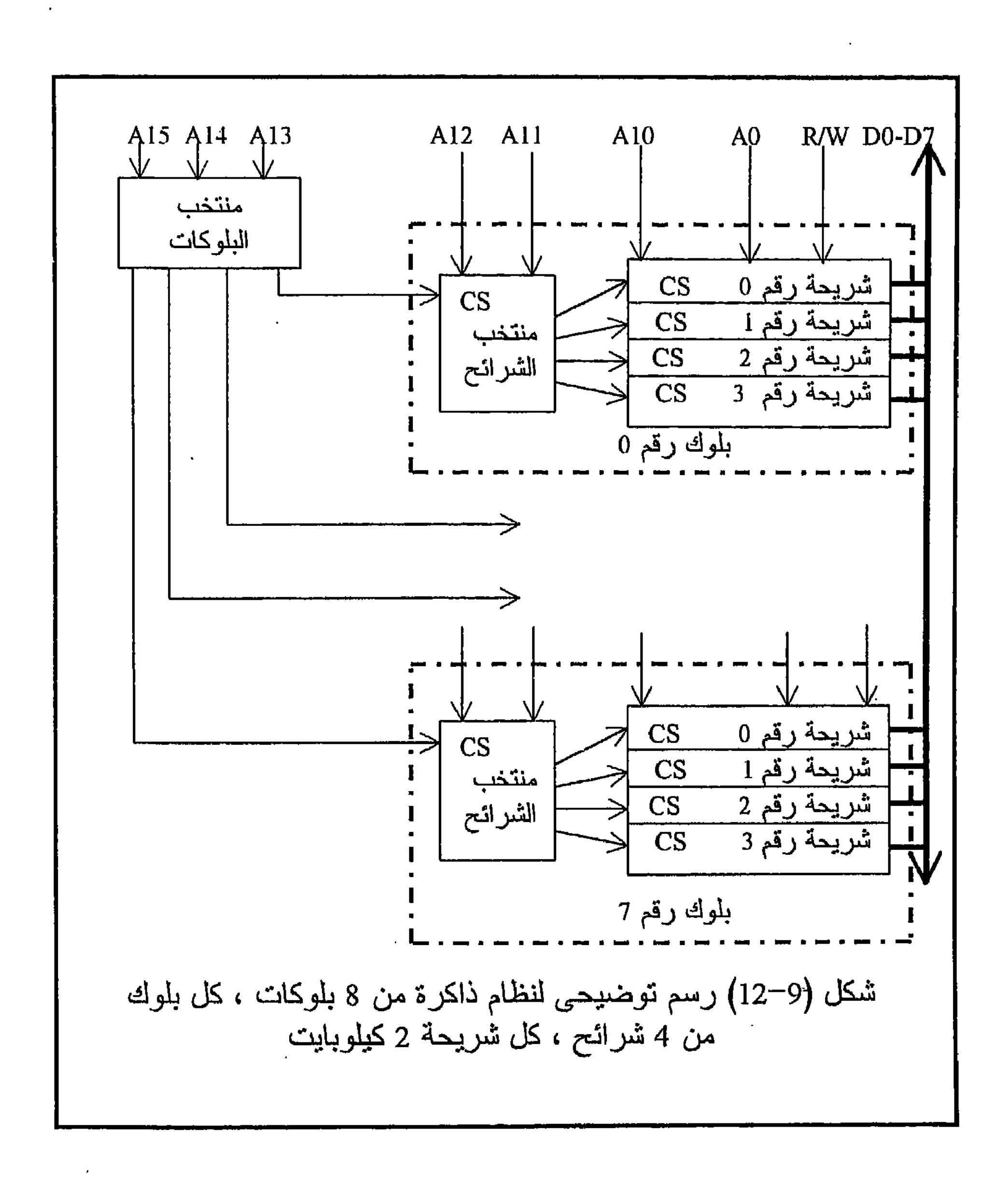
تم اختيارها بإهمال سنترالات المدن . شكل (9-12) يبين رسما توضيحيا لنظام الثمانية بلوكات الذى نفترضه فى شرحنا وذلك حتى يتمكن القارئ من إلقاء نظرة شاملة على نظام الذاكرة بأكمله . لاحظ أن البلوكات ما بين الأول والأخير فى هذا الشكل ما هى إلا تكرار مثل البلوك رقم صفر والبلوك رقم 7 .



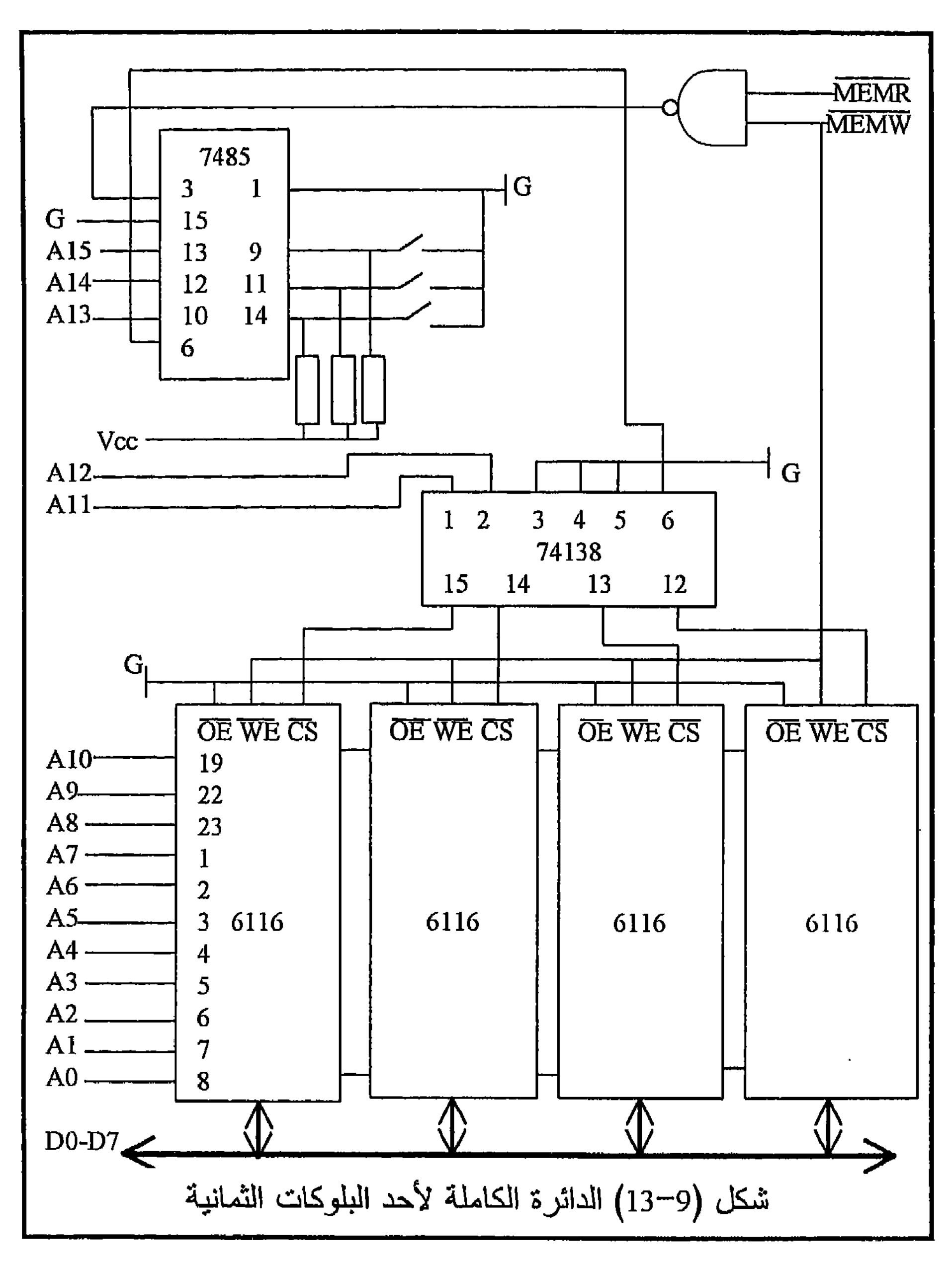
شكل (9-11) نظام ذاكرة مكون من 8 بلوكات وكل بلوك من 4 شرائح

إن نظام بناء الذاكرة باستخدام البلوكات الذى تم شرحه حتى الآن ليس هو النظام الوحيد الذى يجب أن يؤخذ به فى عملية بناء أى نظام ذاكرة ، ولكن من مميزات هذا النظام أنه يعتبر عاما يمكن تعديله ليناسب أى غرض . فمن الممكن مثلا أن يبنى النظام بحيث يكون كل بلوك على كرت منفصل وتكون الذاكرة فى هذه الحالة عبارة عن مجموعة من الكروت التى يستعمل منها ما يلزم للحاجة فقط. أيضا يمكن استخدام نوعى الذاكرة (إما RAM أو ROM) على أى واحد من هذه الكروت ، فقط يجب الحرص عند توصيل خطوط التحكم $\overline{MEMR}$، و $\overline{MEMW}$.
خلاصة القول أن نظام الكروت هذا يمكن أن يحور أو يعدل ليتناسب مع أى تصميم مطلوب . شكل (9-13) يبين الدائرة الكاملة لأحد البلوكات السابقة وقد

استخدمنا معه شريحة الذاكرة رقم 6116 التى تحتوى علـــى 2 كيلوبـــايت RAM استاتيكية . هذه الشريحة متوافقة تماما من حيث وظيفة الأطراف مع الشــــريحة رقم 2716 التى تحتوى على 2 كيلوبايت EPROM بحيث أنه يمكن اســـتخدام أى واحدة من الشريحتين مكان الأخرى .



شكل (9-12) رسم توضيحى لنظام ذاكرة من 8 بلوكات ، كل بلوك من 4 شرائح ، كل شريحة 2 كيلوبايت



شكل (9-13) الدائرة الكاملة لأحد البلوكات الثمانية

إن جميع محتويات هذا الفصل ليست خاصة بمعالج بعينة دون الآخر ولكنها تناسب أى واحد من المعالجات التى هى تحت الدراسة فى هذا الكتاب ، ولقد رأينا فى الفصل السابق كيف حصلنا على المسارات الثلاثة لكل معالج فى الصورة المناسبة لعملية المواجهة ولذلك فإننا نلاحظ أن الشرح لم يكن موجها إلى أى

معالج بعينة ولكن كل ما قيل فى هذا الفصل يمكن تطبيقه مـــع أى واحـــد مـــن المعالجات تحت الدراسة فى هذا الكتاب .

4-9 تمارين

1. مطلوب توصيل شريحة ذاكرة EPROM سعتها 2 كيلوبايت على معالج ، مـــا هى أبسط الطرق لتوصيل هذه الشريحة مع العلم أنها ستكون شـــريحة الذاكـــرة الوحيدة ؟
2. مطلوب توصيل شريحتى ذاكـــرة EPROM و RAM ســـعة كـــل منـــهما 2 كيلوبايت على المعالج ، ما هى أبسط الطرق لذلك مع العلم أنهما شرائح الذاكرة الوحيدة الموصلة على المعالج ؟
3. لديك العديد من شرائح الذاكرة RAM و EPROM التى ســـعة كـــل منـــها 1 كيلوبايت ، ارسم نظام ذاكرة متكامل مكون من بلوكات واقـــترح أنـــت عـــدد البلوكات الذى ستستعمله ؟ اكتب عنوان البداية وعنوان النهاية لكل بلوك وكـــل شريحة ؟
4. ارسم نظام ذاكرة متكامل مكون من 16 بلوك مستخدما شرائح كـــل منـــها 4 كيلوبايت ؟ اكتب عنوان البداية وعنوان النهاية لكل بلوك وكل شريحة؟
5. أعد السؤال الرابع إذا كانت الشرائح المستخدمة سعة كل منـــها 4 كيلوبـــايت أيضا ولكن مسار البيانات لكل منها 4 بتات بدلا من 8 ؟

الفصل العاشر

الإدخال والإخراج

Input and Output

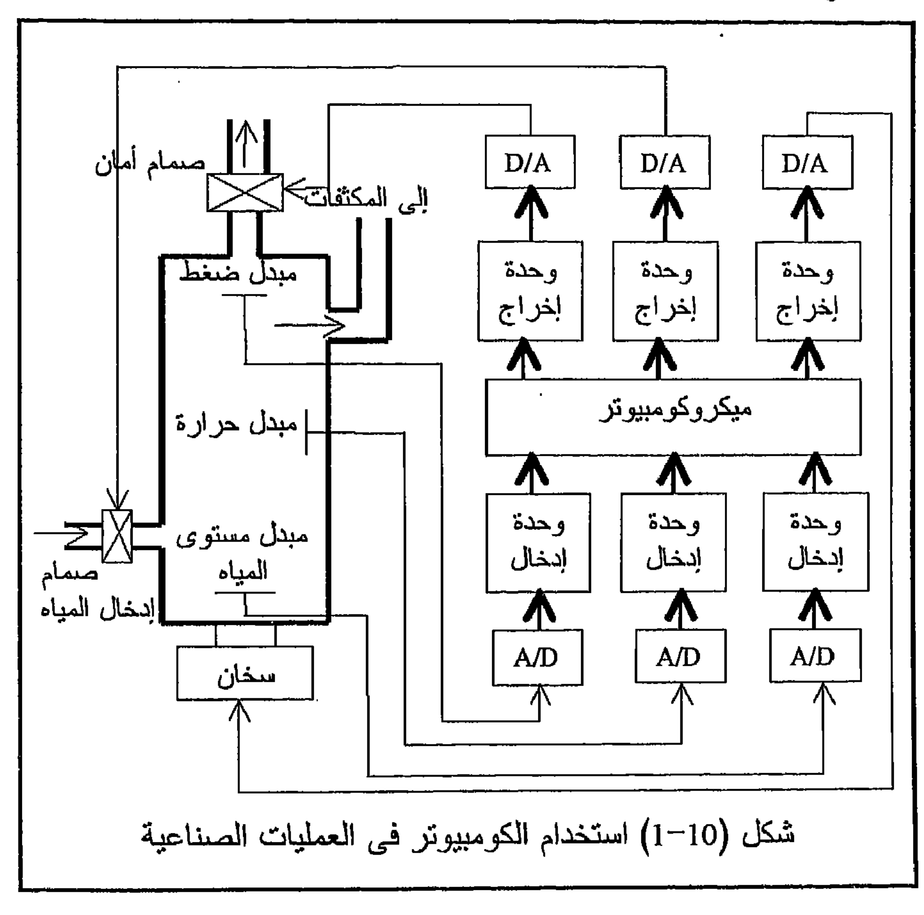
10-1 مقدمة

بالإضافة إلى وحدة التحكم المركـــزى أى ال cpu والذاكـــرة بأنواعـــها فـــإن الكومبيوتر ودوائر التحكم التى تستخدم المعالجات أو الكومبيوتر لابد وأن تحتوى على وحدات لإدخال وإخراج المعلومات والتى من شأنها تسهيل عمليـــة تبـــادل المعلومات بين الأجهزة المحيطة والمعالج . الأجهزة المحيطة مثـــل الطابعـــات والشاشات والمحولات من رقمى إلى تماثلى والعكس وأى أجهزة خارجية يقـــوم المعالج بالتحكم فيها أو التعامل معها . لقد رأينا فى الفصل السابق كيفية توصيـــل المعالج على شرائح الذاكرة سواء كانت RAM أو ROM وسنقوم فى هذا الفصـــل بعرض كيفية بناء وتوصيل المعالج على بوابات الإخراج والإدخال . ونؤكد هنـــا أيضا أن محتويات هذا الفصل ليست خاصة بمعالج بعينه دون الآخـــر ولكنـــها صالحة للاستخدام مع أى واحد من المعالجات التى نستخدمها فى هـــذا الكتـــاب وبالذات بعد أن رأينا فى الفصل الثامن كيفية الحصول على المسارات المختلفـــة للمعالجات وتهيئتها . لكى ننمى الإحساس لدينا بأهميـــة الحاجـــة إلـــى بوابـــات الإخراج والإدخال وأين ومتى تكون الحاجة إليها ماسة سنعرض الآتى :

إن وجود شريحة المعالج كوحدة عمليات مركزية داخل الميكروكومبيوتر ليســـت الوظيفة الوحيدة لمثل هذا النوع مـــن الشـــرائح . هنـــاك مجـــال واســـع مـــن الاستخدامات والتطبيقات لهذه الشرائح وهو استخدامها كعنصـــر أساســـى مـــن عناصر أنظمة التحكم ، فمثلا التحكم فى أى عمليـــة كيماويـــة يمكـــن لشـــرائح المعالجات أن تلعب دورا مهما فيه ، التحكم فى عمليات خلط البـــنزين والـــهواء داخل السيارة مجال من المجالات التى استخدمت هذه الشرائح . إن هذه مجـــرد أمثلة بسيطة ولكن مجمل القول هو أن شرائح المعالجات تلعب فى الوقت الحـــالى دورا مهما جدا فى معظم العمليات الصناعية بل وفى الكثير من الأجهزة والآلات التى نستخدمها فى حياتنا اليومية .

شكل (10-1) يبين نظاما بسيطا لتقطير المياه وقد استخدم الكومبيوتر كعنصـــر أساسى من عناصر التحكم فيه . إن المياه المطلـــوب تقطيرهـــا يتـــم تســـخينها باستخدام سخان إلى أن يتحول الماء إلى بخار فى أعلى التنك حيث يتم سحب هذا البخار فى أنبوبة يمين أعلى التنك حيث يتم تكثيف البخار إلى ميـــاه نقيـــة . إن السخان يجب أن يتوقف عندما تصل درجة حرارة المياه إلى درجة قصوى ويعاد تشغيله إذا وصلت الحرارة إلى حد أدنى ، ويتم ذلك عن طريق ميكروكومبيوتـــر مخزن فيه كل من الدرجتين العظمى والصغرى ويقوم الكومبيوتر بقراءة درجـــة حرارة التنك من خرج مبدل (Transducer) لدرجة الحرارة على فترات زمنيـــة محددة ويقارنها بالدرجتين العظمى والصغرى ويقرر إذا كان سيشغل الســـخان أم يوقفه . بنفس الطريقة يقرأ الكومبيوتر مبدل مستوى المياه فى التنـــك ويقارنـــها

بأعلى وأقل قيمة لمستوى المياه حيث هاتان القيمتان مخزنتان فى الذاكرة ويقـرر بناء على ذلك إذا كان سيتم فتح أو غلق صمام إدخال المياه . إن ضغـط بخـار الماء الموجود فى أعلى التنك من الممكن أن يزيد إلى درجة الخطورة ، لذلك فقد تم وضع مبدل للضغط فى هذا الجزء من التنك حيث يقوم الميكروكومبيوتر مـن وقت لآخر بقراءة خرج هذا المبدل ومقارنتها بقيمتين عظمى وصغـرى وعلـى ضوء هذه المقارنة يقرر إذا كان سيترك صمام الأمان مغلقا أم يفتحه لفترة محددة من الزمن لتقليل الضغـط فـى أعلـى التنـك . مـن ذلـك نـرى أن مهمـة الميكروكومبيوتر هى الاحتفاظ بدرجة حرارة التنك ومستوى المياه فيه وكذلـك مستوى الضغط فى أعلاه عند القيمة المثلى لكل منها . إن عدد المتغيرات التـى يتم التحكم فيها بواسطة الميكروكومبيوتر من الممكن أن يكون أكثر مـن ذلـك بكثير . بل وعادة ما تكون عملية التحكم فى متغيرات تنك المياه هذه عبارة عـن جزء بسيط من عمليات تحكم أخرى يقوم الكومبيوتر بالتحكم فيها داخـل أحـد المصانع .

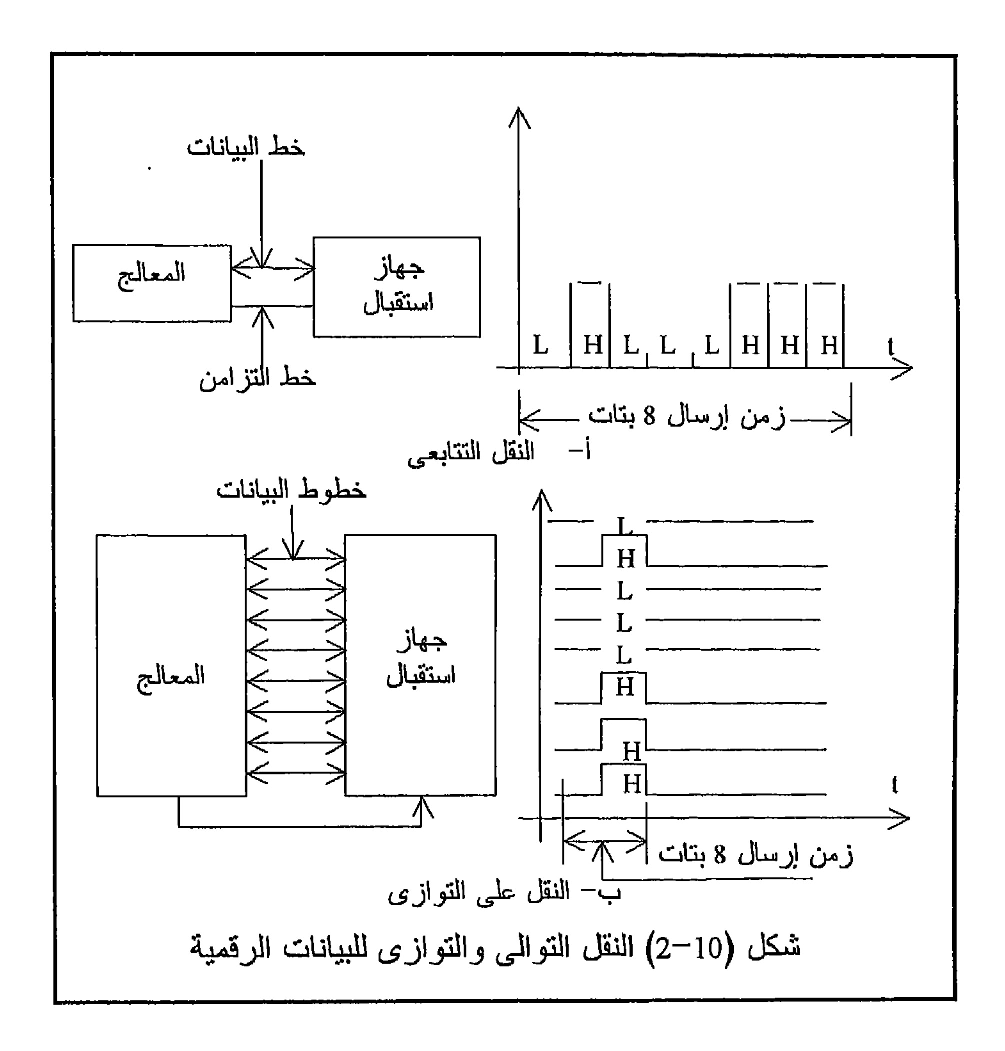


شكل (10-1) استخدام الكومبيوتر فى العمليات الصناعية

إن أهم ما يجب ملاحظته فى هذا المثال الموجود فى شكل (10−1) هو عملية الربط بين عالم الرقميات المتمثل فى الكومبيوتر حيث تكون جميع الإشارات الموجودة رقمية فقط (Digital) ، وعالم التناظريات أو التماثليات المتمثل فى تنك المياه وإشارات التحكم فيه حيث تكون جميع هذه الإشارات تماثلية (Analog) . فمثلا لكى يعمل السخان لابد من توصيله بجهد عال (220 فولت مثلا) ، وكذلك لكى تفتح أو تغلق أيا من الصمامات لابد من استخدام هذا الجهد العالى ، إذن كيف يتم هذا الربط ؟ أو بمعنى آخر كيف تحول الإشارات الرقمية الخارجة من الكومبيوتر إلى إشارات تماثلية أو جهد عال يتم به تشغيل السخان مثلا؟ وكيف يتم أخذ الإشارة الخارجة من أى من المبدلات وهى إشارة تماثلية غالبا ما تكون ضعيفة ، كيف يتم تحويلها إلى إشارة رقمية يمكن للكومبيوتر أن يتعامل معها ؟ جميع هذه الأسئلة وغيرها سيتم الإجابة عنها فى هذا الفصل والفصول القادمة . قبل الإجابة عن هذه الأسئلة سنبذل بعض الوقت فى التعرف على كيفية إدخال وإخراج المعلومات إلى ومن الكومبيوتر بافتراض أن هذه المعلومات موجودة فى الصورة الرقمية .

10−2 طرق إرسال واستقبال المعلومات الرقمية

إن المعلومات التى يتم إدخالها أو إخراجها من أو إلى الكومبيوتر تكون إما فى صورة تتابعيه Serial data أو فى صورة متوازية Parallel data وكل من الصورتين له مميزاته والاستخدامات الخاصة التى تحتاج إليه بالضرورة . فى الطريقة التتابعية يتم إرسال المعلومات من وإلى الأجهزة الخارجية على خط واحد فقط ولا يرسل على هذا الخط إلا بت bit واحدة فقط فى نفس وحدة الزمن وهى ال Clock ، بحيث أنه لكى يتم إرسال معلومة من ثمانية بتات مثلا فإننا نحتاج إلى زمن مقداره ثمانى نبضات تزامن لكى نرسل هذه المعلومة . أما فى الطريقة المتوازية فإن المعلومات يتم إرسالها من وإلى الكومبيوتر على أكثر من خط واحد ، وعادة ما يكون عدد هذه الخطوط يساوى عدد الخطوط فى مسار البيانات للكومبيوتر الذى يتم التعامل معه وفى هذه الحالة فإننا لكى نرسل معلومة من ثمانى بتات مثلا سنحتاج إلى ثمانية خطوط متوازية بحيث ترسل كل بت على خط منفصل من هذه الخطوط وبالطبع فإنه فى هذه الحالة سترسل جميع هذه البتات فى خلال نبضة تزامن Clock واحدة فقط ، لذلك فإن هذه الطريقة أسرع بكثير فى إرسال المعلومات من الطريقة التتابعية السابقة . شكل (10−2) يبين رسما توضيحيا لطرق إرسال المعلومات بالطريقتين التتابعية والمتوازية .



شكل (2-10) النقل التوالى والتوازى للبيانات الرقمية

إن عملية إدخال أو إخراج معلومة من أو إلى الكومبيوتر تتكون من جزأين ، الجزء الأول منها هو برنامج Software يقوم الكومبيوتر بتنفيذه ، والجزء الثانى هو دائرة Hardware يتم بناؤها لكى تقوم بدور الوسيط بين الكومبيوتر والأجهزة الخارجية . لكتابة برنامج يقوم بإدخال أو إخراج معلومات من أو إلى الكومبيوتر فإن هناك أيضا طريقتين : الطريقة الأولى يتم فيها استخدام الأمر IN لإدخال المعلومة والأمر OUT لإخراجها والتى تسمى بطريقة خريطة الإدخال والإخراج للإدخال والإخراج Input Output mapped I/O . الطريقة الثانية وفيها يتم استخدام خريطة ذاكرة الكومبيوتر ولذلك تسمى هذه الطريقة Memory mapped I/O حيث يعامل الجهاز الخارجى كما لو كان بايت Byte من بايتات الذاكرة ولذلك يمكن مع هذه الطريقة استخدام أوامر المعالج العادية مثل الأوامر STA, LDA, MOV, وذلك على العكس من الطريقة الأولى والتى لا يستخدم معها إلا

الأمرين IN, OUT فقط . سيأتى شرح هذه الطريقة فيما بعد وكذلك سنرى أن الدائرة أو ال Hardware الذى يستخدم مع كل من الطريقتين لا يختلف كثيرا .

3-10 الطريقة الأولى من طرق الإدخال والإخراج باستخدام الأمرين OUT, IN

هذه الطريقة من طرق الإدخال والإخراج لا تستعمل مع المعالج MC6800 على الإطلاق حيث لا تحتوى قائمة أوامر هذا المعالج على الأمرين IN و OUT ولكن تستخدم معه طريقة خرائط الذاكرة التى سنشرحها فى الجزء القادم إن شاء الله .

كما نعلم فإن الأمرين OUT, IN يتكون كل منهما من 2 بايت ، البايت الأولى هى شفرة الأمر ، أما البايت الثانية فتحتوى على رقم بوابة الإخراج أو الإدخال التى سيتم التعامل معها . كما نعلم أيضا فإن البايت تتكون من ثمانى بتات ولذلك فإنه يمكن تشفير عدد من بوابات الإدخال أو بوابات الإخراج يصل إلى 256 بوابة (2^8) . عندما يقوم المعالج بتنفيذ الأمر OUT أو الأمر IN فإن رقم البوابة يتم وضعه على الثمانية خطوط الأولى من مسار العناوين ، فمثلا عند تنفيذ الأمر OUT 05 حيث 05 هى رقم أو عنوان بوابة الإخراج فإن المعالج يضع الرقم 05 على مسار العناوين كالتالى :

A7	A6	A5	A4	A3	A2	A1	A0
0	0	0	0	0	1	0	1

أما محتويات مسجل التراكم وهى المعلومة المطلوب إخراجها فيتم وضعها على مسار البيانات تمهيدا لالتقاطها بواسطة بوابة الإخراج فى اللحظة المناسبة . كذلك الحال بالنسبة للأمر IN فمثلا عند تنفيذ الأمر IN 3F فإن رقم بوابة الإدخال وهو 3F يقوم المعالج بوضعه على مسار العناوين كالتالى :

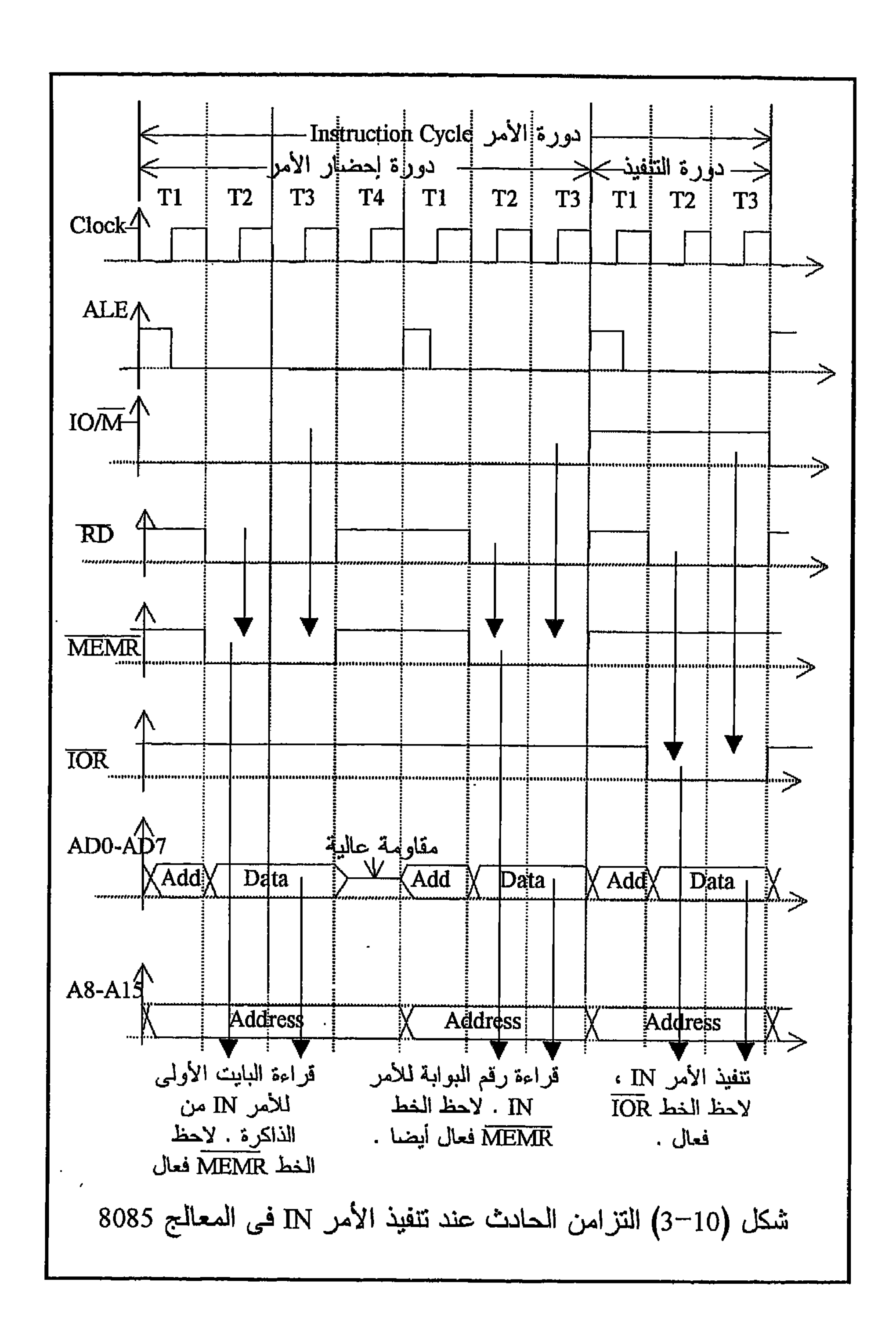
A7	A6	A5	A4	A3	A2	A1	A0
0	0	1	1	1	1	1	1

أما المعلومة المراد إدخالها فتنتقل من مسار البيانات الذى يكون متصلا ببوابة الإدخال فى هذه اللحظة إلى مسجل التراكم ، ويجب التأكيد هنا على أنه فى أثناء تنفيذ الأمر OUT فإن الخط $\overline{IOW}$ من مسار التحكم يكون فعالا أى صفرا ، وكذلك فى أثناء تنفيذ الأمر IN فإن الخط $\overline{IOR}$ من مسار التحكم يكون فعالا ويساوى أيضا صفرا . لذلك فإن الخطين $\overline{IOW}$ و $\overline{IOR}$ مهمان جدا فى بناء كل من دائرتى الإدخال والإخراج كما سنرى ، ولقد رأينا فى فصل سابق كيفية الحصول على هذه الخطوط مع كل معالج نقوم بدراسته فى هذا الكتاب .

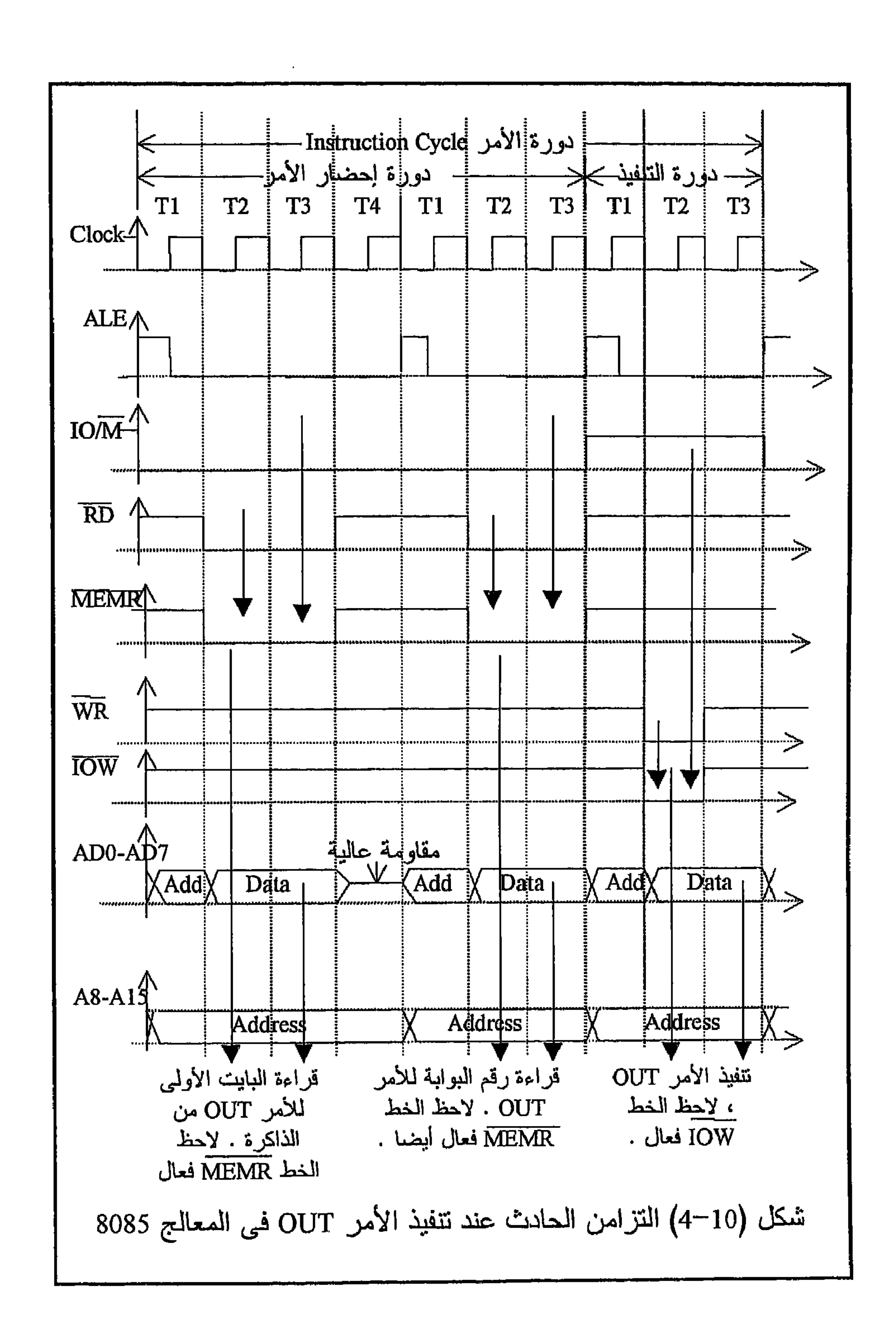
الشكلان (10–3) و (10–4) يبينان التزامن الموجود على جميع خطوط التحكم التى تتأثر بالأمرين IN و OUT فى حالة المعالج 8085 . هذان الشكلان يستحقان التأمل والتدقيق من قبل أى قارئ حيث أنه يمكن منهما ملاحظة وفهم الكثير من طريقة تنفيذ المعالج لأى أمر والتزامن الذى يحدث بين إشارات التحكم المختلفة . شكلا (10–5) و (10–6) يبينان التزامن الموجود على جميع خطوط التحكم التى تتأثر بالأمرين IN و OUT فى حالة المعالج Z80 وهذان الشكلان يستحقان التدقيق أيضا من جميع القراء لملاحظة مدى التشابه بين معظم هذه الإشارات .

10–3–1 دائرة بوابة الإخراج Output port

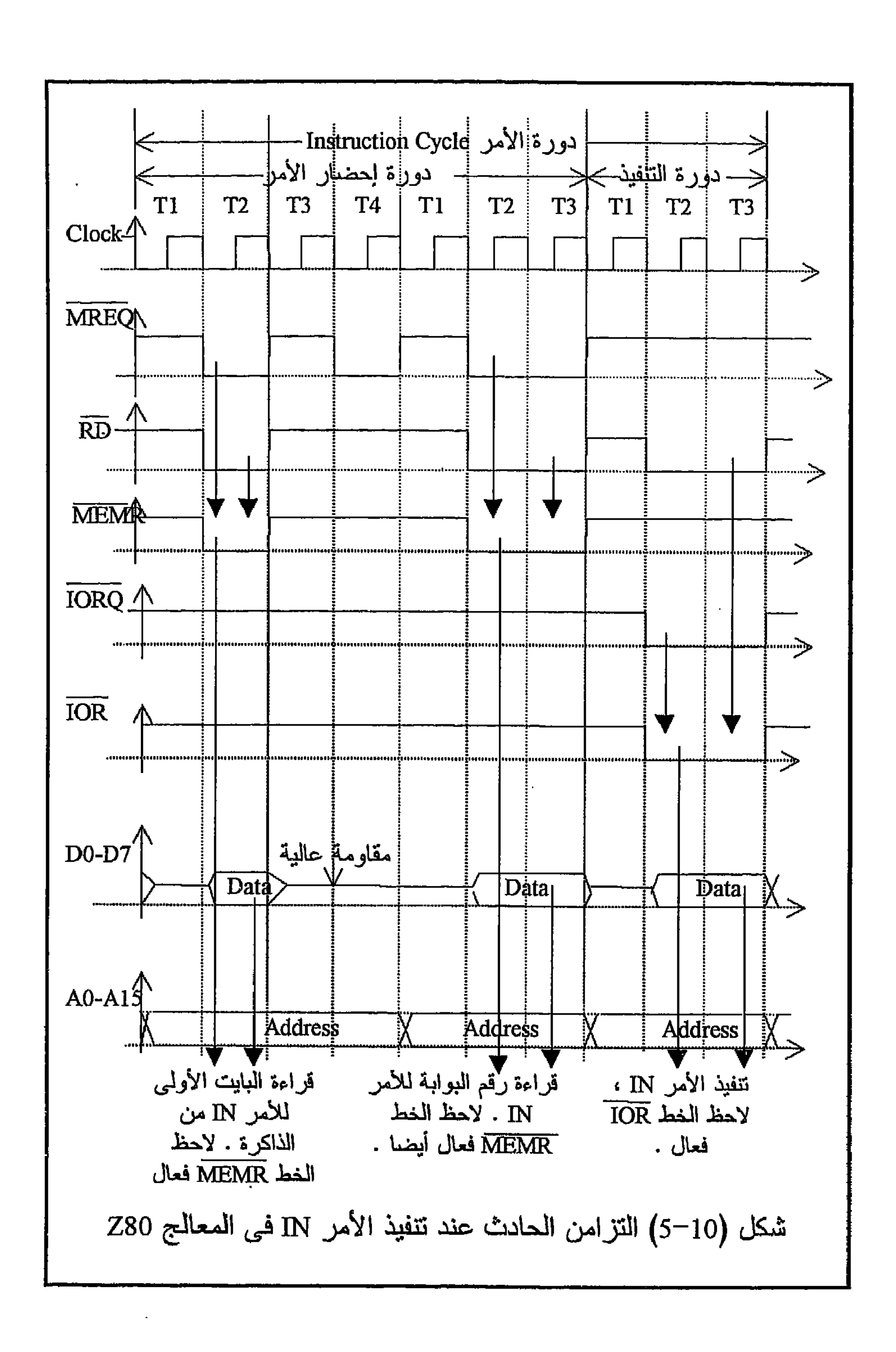
شكل (10–7) يبين رسما تخطيطيا لبوابة إخراج . كما رأينا فإن كلا من الشريحة Intell 8085 و Z80 يمكنها التعامل مع 256 بوابة إخراج يمكن ترقيمها من البوابة رقم صفر إلى البوابة رقم 255 . لذلك فقد خصصت المفاتيح S1 إلى S8 لاختيار رقم للبوابة ووضعه فى الصورة الثنائية باستخدام هذه المفاتيح . عند تنفيذ الأمر OUT فإن رقم البوابة كما ذكرنا من قبل يظهر على الثمانية خطوط الأولى من مسار العناوين A0 إلى A7 . هنا تكون مهمة مقارن العناوينAddress Comparator حيث يقارن الرقم الموجود على مسار العناوين مع الرقم الموضوع بالمفاتيح ، فإذا تطابق الرقمان يقوم مقارن العناوين بإعطاء إشارة خرج تدل على ذلك ، وإذا لم يتطابقا يظل خرجه خاملا دون تغيير . تستخدم إشارة الخرج القادمة من مقارن العناوين لتشغيل ماسك الخرج Output latch ذى الثمانية بتات كما هو مبين بشكل (10–7) . لاحظ أن دخل ماسك الخرج متصل بمسار البيانات والذى توضع عليه المعلومة المراد إخراجها فى أثناء تنفيذ الأمر OUT ، وفى حالة وجود إشارة الخرج القادمة من مقارن العناوين يقوم ماسك الخرج بنقل المعلومة الموجودة على دخله إلى خرجه حيث يمكن إظهارها على شاشة أو مظهرات أو الاستفادة منها فى أى غرض من الأغراض . الذى يهمنا الآن هو أن المعلومة الموجودة داخل المعالج وبالتحديد فى مسجل التراكم تم إخراجها على ماسك حيث من هنا تبدأ الاستفادة بها . لاحظ أيضا أن الخط $\overline{IOW}$ تم استخدامه كخط تشغيل مع مقارن العناوين بحيث لا يعمل مقارن العناوين إلا إذا كان هذا الخط فعالا أى يساوى صفرا .



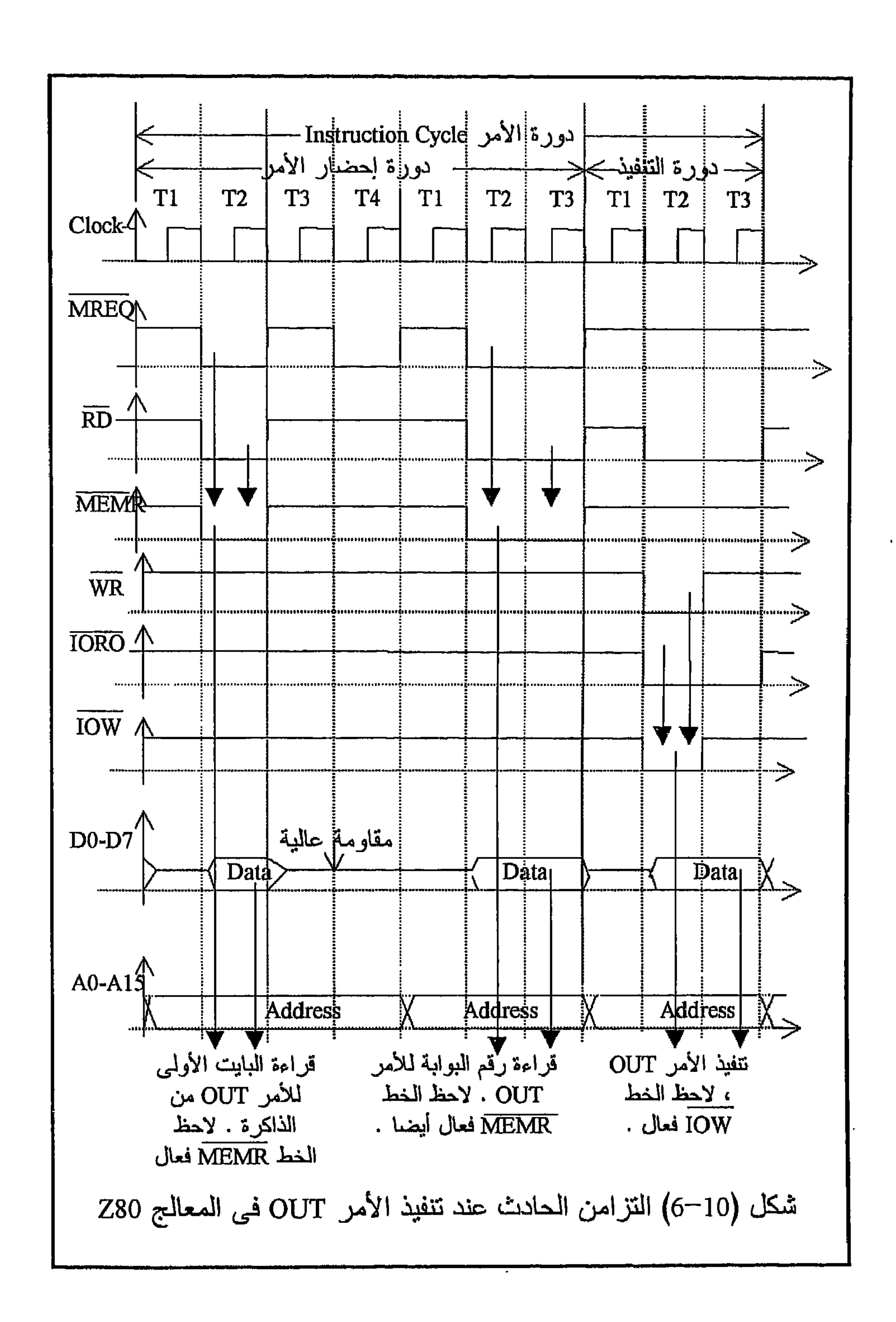
شكل (3-10) التزامن الحادث عند تنفيذ الأمر IN فى المعالج 8085



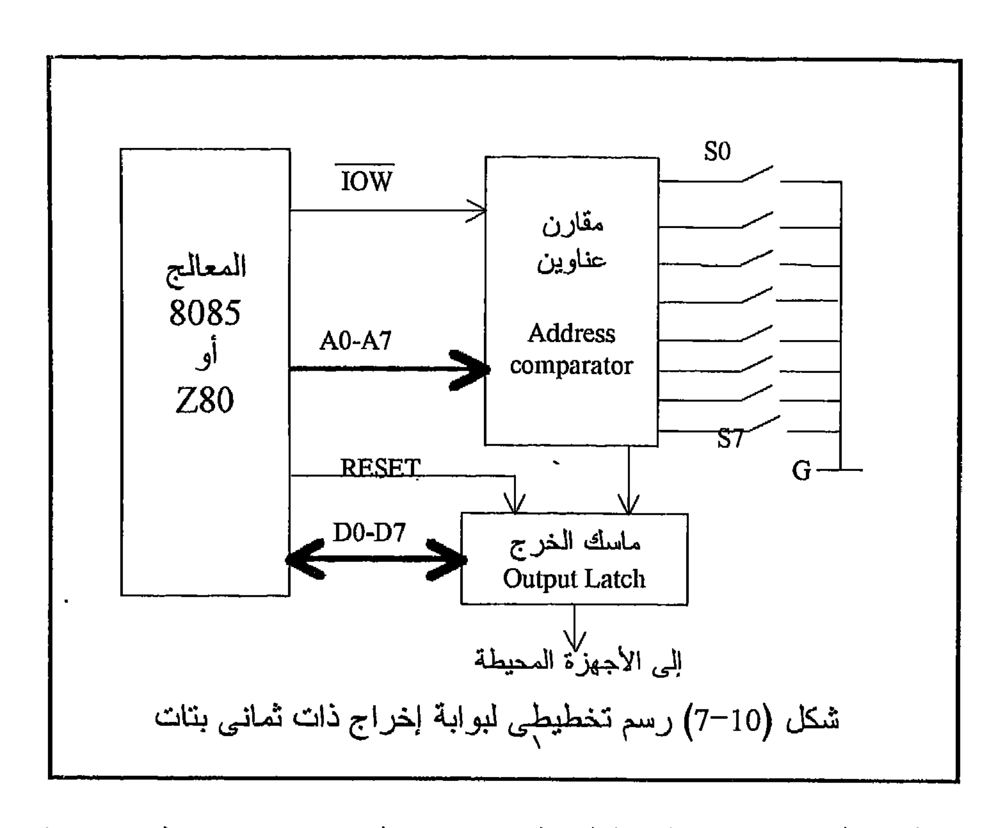
شكل (4-10) التزامن الحادث عند تنفيذ الأمر OUT فى المعالج 8085



شكل (5-10) التزامن الحادث عند تنفيذ الأمر IN فى المعالج Z80



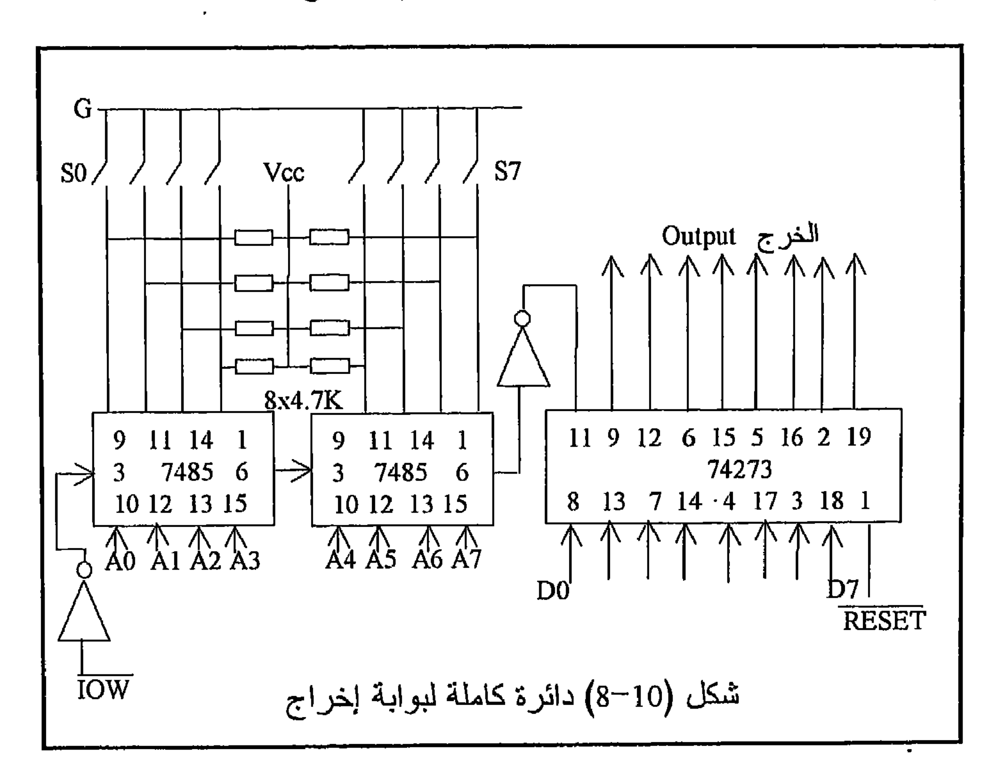
شكل (6-10) التزامن الحادث عند تنفيذ الأمر OUT فى المعالج Z80



شكل (10−7) رسم تخطيطى لبوابة إخراج ذات ثمانى بتات

شكل (10−8) يبين الدائرة الكاملة لبوابة الإخراج . الشريحتان 7485 كل منهما عبارة عن مقارن ذى أربع بتات حيث يكون خرج الشريحة على الطرف 6 يساوى High إذا كانت كل بت من الدخل A تساوى نظيرتها من الدخل B وكان دخل الشريحة على الطرف 3 يساوى High أيضا . لذلك فإن الإشارة $\overline{IOW}$ تم إدخالها على الطرف 3 للشريحة الأولى وتم توصيل خرج هذه الشريحة (الطرف 6) إلى دخل الشريحة الثانية (الطرف3) ، بذلك تعمل الشريحتان كمقارن ذى ثمانى بتات يؤخذ خرجه من الطرف 6 للشريحة الثانية . هذا الخرج من الشريحة الثانية يستخدم كنبضات تزامن Clock للشريحة 74273 وهى الماسك وذلك بعد عكسه . الشريحة 74273 عبارة عن ماسك ذى ثمانى بتات وعند عبور التزامن (الطرف 11) من Low إلى High تنتقل الإشارة الموجودة على الدخل إلى الخرج مباشرة . الطرف 1 للشريحة 74273 خاص بالتصفير Clear أو CLR بحيث عندما يكون هذا الطرف Low يصبح كل خرج الشريحة يساوى صفر ولذلك فقد تم توصيل هذا الطرف على ال RESET لضمان أن يكون خرج الماسك يساوى صفرا عند بداية التشغيل . لاحظ أن الاستفادة من الخرج تبدأ من هنا حيث يمكن توصيل خرج الماسك على محول رقمى/تماثلى أو على مظهر وذلك على حسب

التطبيق الذى ستستخدم فيه هذه البوابة . لقد كان من الممكن استخدام منتخب decoder بدلا من مقارن العناوين ولكن ميزة استخدام المقارن أنه يمكن تغيير رقم أو عنوان البوابة على حسب الطلب باستخدام المفاتيح S0 إلى S7 .

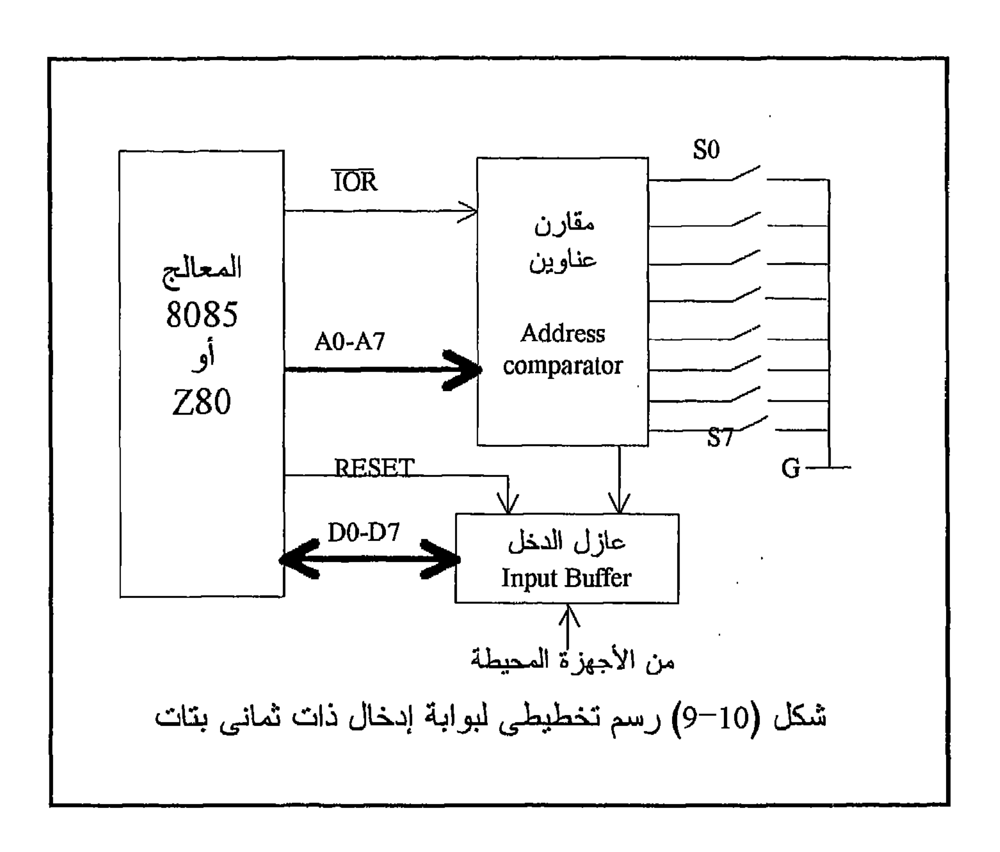


شكل (10-8) دائرة كاملة لبوابة إخراج

10-3-2 دائرة بوابة الإدخال Input port

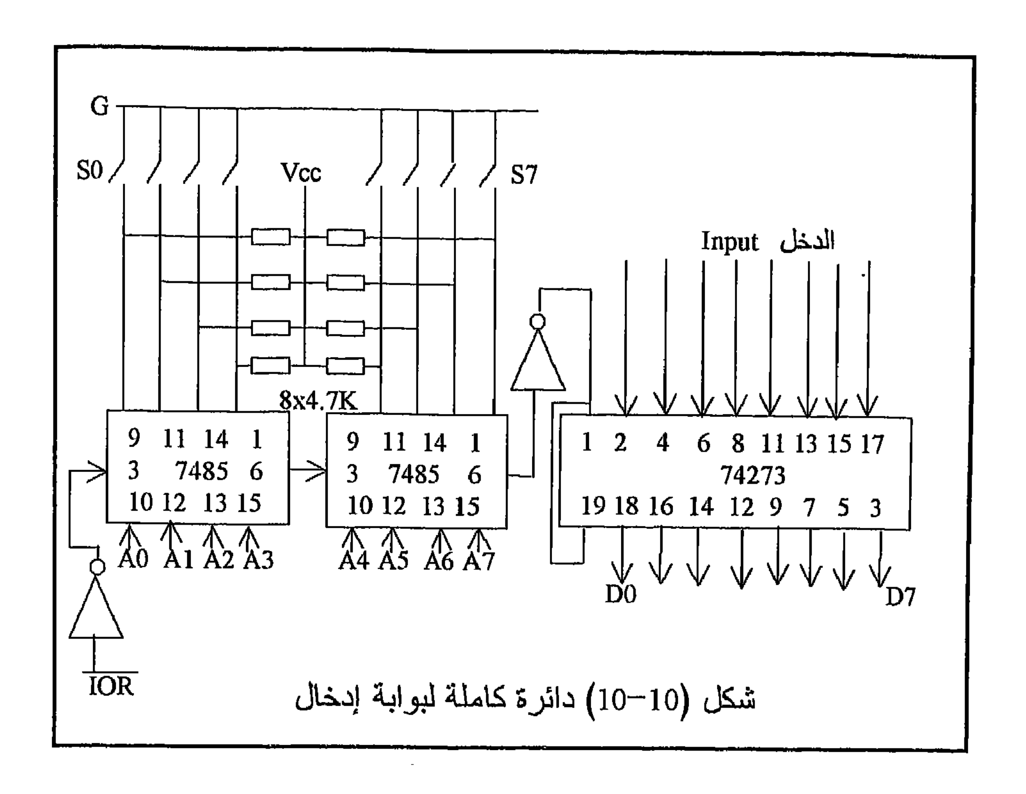
شكل (10-9) يبين رسما تخطيطيا لبوابة إدخال . كما علمنا فإن كلا من الشريحة Intell 8085 و Z80 يمكنهما التعامل مع 256 بوابة إدخال والتى يمكن ترقيمها أو عنونتها من البوابة رقم صفر حتى البوابة رقم 255. لذلك فقد خصصت المفاتيح S1 حتى S8 لاختيار رقم للبوابة ووضعه فى الصورة الثنائية باستخدام هذه المفاتيح . عند تنفيذ الأمر IN فإن رقم البوابة كما ذكرنا من قبل يظهر على الثمانية خطوط الأولى من مسار العناوين A0 إلى A7 ، لذلك فإن مهمة مقارن العناوين تكون هى مقارنة الرقم الموجود على مسار العناوين مع الرقم الموضوع بالمفاتيح ، فإذا تطابق الرقمان يعطى إشارة خرج وإذا لم يتطابقا يظل الخرج خاملا أى غير فعال . تستخدم إشارة الخرج القادمة من مقارن العناوين لتشغيل عازل Buffer ذى ثمانى بتات الذى رقمه 74244 ، انظر إلى محتويات هذه الشريحة فى الفصل الثامن . إن دخل العازل متصل بالمصدر

الذى تأتى منه المعلومة المراد إدخالها إلى المعالج ، لاحظ أنـــه عندمـــا يكـــون مقارن العناوين خاملا أى أن خرجه غير فعال فإن العازل أيضا ســـيكون غـــير فعال وسيكون خرجه عبارة عن مقاومة عالية حتى لا يســـبب تحميـــلا لمســـار البيانات Short circuit فى حالة اتصال مسار البيانات بأى جهاز آخر ولقد عرفنـــا ذلك فى دراستنا لبوابات المنطق الثلاثى . لقد تم استخدام الخـــط $\overline{IOR}$ كخـــط تشغيل مع مقارن العناوين بحيث لا يعمل مقارن العناوين إلا إذا كان هذا الخـــط فعالا أى يساوى صفرا .



شكل (10-9) رسم تخطيطى لبوابة إدخال ذات ثمانى بتات

شكل (10-10) يبين الدائرة الكاملة لبوابة إدخال . لاحظ مدى التناظر بين هـــذه الدائرة ودائرة بوابة الإخراج . الفرق الأساسى هو استخدام ماسك للخرج فـــى حالة بوابة الإخراج لمسك المعلومة التى سيتم إخراجها أمـــا فـــى حالـــة بوابـــة الإدخال فيستخدم عازل Buffer لعزل مسار البيانات عن المصدر الذى تدخل منه المعلومة .



شكل (10−10) دائرة كاملة لبوابة إدخال

10−4 الطريقة الثانية من طرق الإخراج والإدخال باستخدام خريطة الذاكرة Memory mapped I/O

إن خريطة الذاكرة لأى حاسب هى عبارة عن رسم تخطيطى يبين فيم تستخدم كل بايت من بايتات الذاكرة ابتداء من أول بايت إلى آخر بايت فى الذاكرة . فمثلا خريطة الذاكرة الخاصة بالميكروكومبيوتر الذى نستخدمه مع أمثلة هذا الكتاب والموضحة فى شكل (10−11) فيها البايتات ابتداء من 0000 إلى 1FFF مشغولة بذاكرة قراءة فقط EPROM , Read only memory , أما نظام التشغيل الخاص بهذا الميكروكومبيوتر فيشغل الذاكرة ابتداء من البايت FC00 إلى FFFF. انظر إلى شكل (10−11) لتتعرف فيما تستخدم باقى الذاكرة . يمكن كما سنرى استخدام طريقة خرائط الذاكرة للإدخال والإخراج مع أى معالج ولكن مع المعالج MC6800 فإن هذه الطريقة هى الوحيدة التى يمكن استخدامها معه وذلك لأن هذا المعالج ليس لديه أوامر إدخال وإخراج مثل أبناء جيله 8085 و Z80 .
فى هذه الطريقة من طرق الإدخال والإخراج تعامل بوابة الإدخال أو الإخراج كما لو كانت بايت من بايتات الذاكرة ولذلك فإن جميع أوامر المعالج الخاصة

بالتعامل مع الذاكرة يمكن استخدامها فى هذه الحالة . لكى نستخدم بـــايت مــن بايتات الذاكرة فى هذا الغرض فإنه يجب التأكد من أن هذه البايت غير مستخدمة فى أى غرض من أغراض التخزين الأخرى ويمكن معرفة ذلك طبعا بعد النظـر فى خريطة الذاكرة وانتقاء البايت المناسبة لذلك بحرص شـــديد . بـــالنظر فـــى خريطة الذاكرة فإننا سنبتعد عن مساحات الذاكرة المســـتخدمة لنظـــام التشـــغيل والمستخدمة كذاكرة قراءة فقط وكذلك المستخدمة كذاكرة تخزيـــن Read/Write memory وسنلجأ للمساحة المحددة بالعنوانين A000 إلى BFFF مثـــلا وســـوف نستخدم أى مكانين فى هذه المساحة أحدهما كبوابة إدخال والآخر كبوابة إخراج.
إنه فى هذا المجال يجب أن نذكر جيدا أنه عندما تقوم شريحة المعالج بتنفيـــذ أى أمر من الأوامر التى تتعامل مع الذاكرة فإن عنوان المكان الذى ســـيتم التعـــامل معه يوضع على مسار العناوين (16 خطا) وفى هذه الحالة فإن خطـــى التحكـــم $\overline{MEMR}$ و $\overline{MEMW}$ يوضحان ما إذا كان هذا التعامل سيكون بغرض القـــراءة من الذاكرة أو الكتابة فيها على حسب ما سيكون أى من الخطين فعـــالا . لذلـــك فإن كلا من هذين الخطين سيلعبان دورا مهما فى عملية التشفير الخاصة ببوابـــة الإدخال أو بوابة الإخراج كما سنرى فيما يلى :

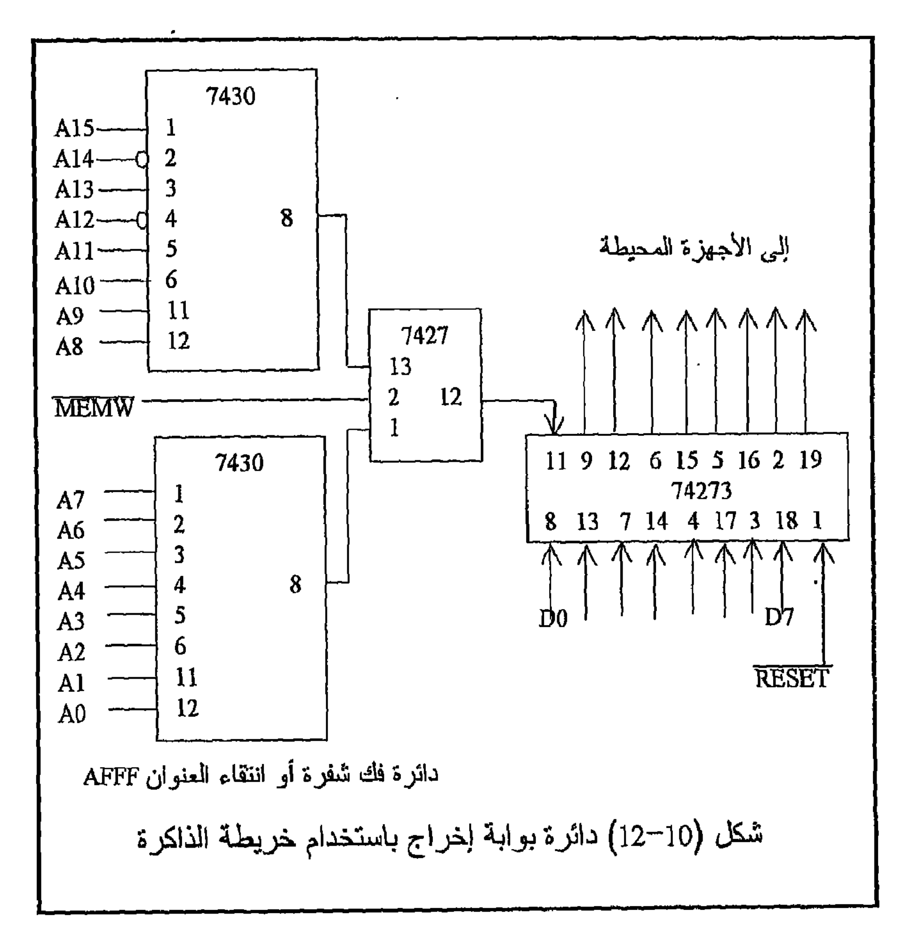
العنوان	المحتوى
0000 – 1FFF	Read Only Memory ROM
2000 – 3FFF	Free
4000 – E000	Free
E000 – FE00	RAM
FE00 – FFFF	محجوز لنظام التشغيل

شكل (10-11) مثال لخريطة الذاكرة لأحد الأنظمة

10-4-1 دائرة بوابة إخراج باستخدام خريطة الذاكرة

افترض مثلا أنه بالنظر فى خريطة الذاكرة الخاصة بـــالميكروكومبيوتر الـــذى نستخدمه وجدنا أن مساحة الذاكرة التى تبدأ من المكان A000 إلى المكان BFFF غير مستخدمة لأى غرض من أغراض التخزين ، لذلك فإننا سنســتخدم أحــد أماكن هذه المساحة كبوابة إخراج وليكن مثلا المكان رقم AFFF . شــكل (10-12) يبين دائرة مقترحــة لهذه البــوابة . عند تنفيذ الأمرين :

```
MVI  A,05
STA  AFFF
```



شكل (10-12) دائرة بوابة إخراج باستخدام خريطة الذاكرة

بواسطة المعالج فإن الأمر الأول سيضع الرقم 05 فى مسجل التراكم والأمر الثانى سيتسبب فى وضع العنوان AFFF على مسار العناوين (A0 إلى A15) وسيجعل الخط $\overline{MEMW}$ فعالا ، ثم يضع محتويات مسجل التراكم على مسار البيانات ، لذلك فإنه بالنظر إلى شكل (10−12) فإن خرج فاكك الشفرة سيكون فعالا مما يسبب نبضة تزامن Clock pulse للماسك (الشريحة 74273) الذى يقوم بإخراج محتويات مسار البيانات على الخرج وذلك هو المطلوب من أى بوابة إخراج حيث يمكن فى هذه الحالة الاستفادة من الخرج أو إظهاره على شاشة أو مظهر من أى نوع كما فعلنا مع بوابة الإخراج سابقا . لاحظ أنه يمكن استخدام مقارن للعناوين كما فى الأجزاء السابقة بدلا من فاكك الشفرة أو منتقى العنوان AFFF الموجود فى شكل (10−12) .

10−4−2 دائرة بوابة إدخال باستخدام خريطة الذاكرة

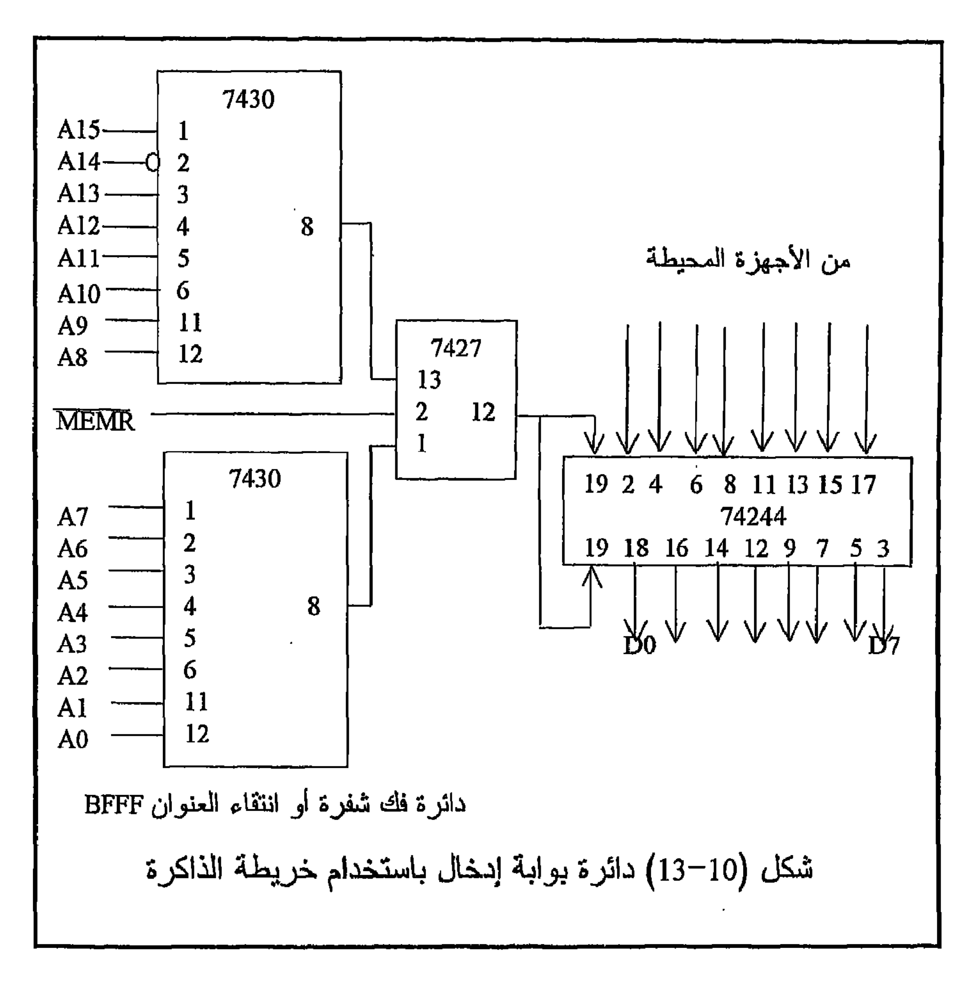
افترض أننا سنستخدم المكان BFFF من أماكن الذاكرة كبوابة إدخال . شكل (10−13) يبين الدائرة المقترحة لذلك . عندما تقوم شريحة المعالج بتنفيذ الأمر:

LDA BFFF

فإن العنوان BFFF سيوضع على مسار العناوين (A0 إلى A15) وسيكون الخط $\overline{MEMR}$ فى هذه المرة فعالا ، لذلك فإنه فى هذه الحالة سيكون خرج فاكك الشفرة أو المنتقى فعالا كما فى شكل (10−13) مما سيسبب نبضة تزامن لشريحة العازل Buffer والتى بسببها يقوم العازل بنقل المعلومة الموجودة على دخله وهى المعلومة القادمة من الخارج ويضعها على مسار البيانات حيث يقوم المعالج بنقل هذه المعلومة إلى مسجل التراكم . لاحظ أن العازل Buffer لابد وأن يكون من النوع ثلاثى المنطق .

بالنظر إلى دوائر الإدخال والإخراج باستخدام خريطة الذاكرة وباستخدام الأمرين IN, OUT نلاحظ مدى التناظر بينهما ، ففى أى من الطريقتين لابد وأن يكون هناك فاكك شفرة أو منتقى لرقم البوابة ، الاختلاف هو أنه فى حالة استخدام خريطة الذاكرة فإننا نشفر مسار العناوين بالكامل (16 خطا) وذلك فى حالة التشفير الكامل ، أما فى حالة استخدام الأمرين IN, OUT فإننا نشفر الثمانية خطوط الأولى فقط من مسار العناوين وذلك أيضا فى حالة التشفير الكامل . هناك أيضا فرق أساسى وهو أننا مع طريقة خريطة الذاكرة نستخدم خطى التحكم $\overline{MEMR}$ و $\overline{MEMW}$ أما مع الطريقة الأخرى فإننا نستخدم الخطين $\overline{IOR}$ و $\overline{IOW}$. فيما عدا ذلك فإن الماسك Latch أو العازل Buffer يستخدمان فى كل من الطريقتين إما لمسك معلومة الخرج فى حالة بوابة الإخراج أو لعزل مصدر المعلومة عن مسار البيانات فى حالة بوابة الإدخال .

لقد راعينا فى جميع الدوائر السابقة استخدام عملية التشفير الكاملة للبوابات فـــى الحالتين سواء باستخدام الأمرين IN, OUT أو باستخدام خرائط الذاكرة ، ونعنـــى بالتشفير الكامل أن جميع خطوط العناوين (الثمانية أو الستة عشر علـــى حســب الطريقة المستخدمة) تستخدم فى عملية التشفير ، فمـــع خرائـــط الذاكـــرة مثـــلا استخدمنا جميع خطوط مسار العناوين (A0 إلى A15) وفى حالة الأمريـــن IN, OUT استخدمنا ال 8 خطوط الأولى من مسار العناوين . فى الحقيقة فإن دائـــرة التشفير من الممكن أن تكون أبسط من الدوائر السابقة بكثير لو أننـــا اســـتخدمنا عددا أقل من خطوط العناوين .



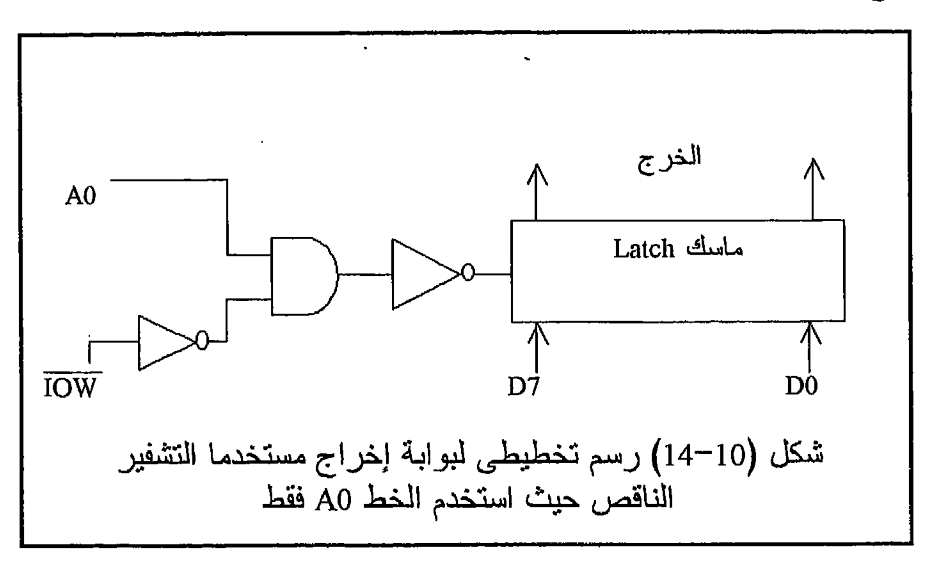
شكل (10–13) دائرة بوابة إدخال باستخدام خريطة الذاكرة

شكل (10–14) يبين عملية تشفير لبوابة إخراج باستعمال الأمر OUT وباســتخدام خط واحد من مسار العناوين وهو الخط A0 . المشكلة مع مثل هذه البوابة هى أن

أى رقم أو عنوان تكون فيه البت A0 تساوى واحدا سيتسبب فى تشغيلها . فمثـــلا جميع الأوامر التالية تصلح لتشغيل هذه البوابة :

OUT 01
OUT 03
OUT FF

وهكذا فإن أى رقم يبدأ بواحد فى البت A0 سيشغل هذه البوابة . فى الحقيقة فـإن بناء مثل هذه البوابات الناقصة التشفير ليس به أى عيب طالما أنه ليست هنـــاك بوابات تحمل هذه الأرقام المتكررة ويجب أن يراعى ذلك فى عملية التصميـــم . نفس الكلام يمكن تطبيقه على عمليات تشفير البوابات التـــى تســتخدم خرائــط الذاكرة .



شكل (10-14) رسم تخطيطى لبوابة إخراج مستخدما التشفير الناقص حيث استخدم الخط A0 فقط

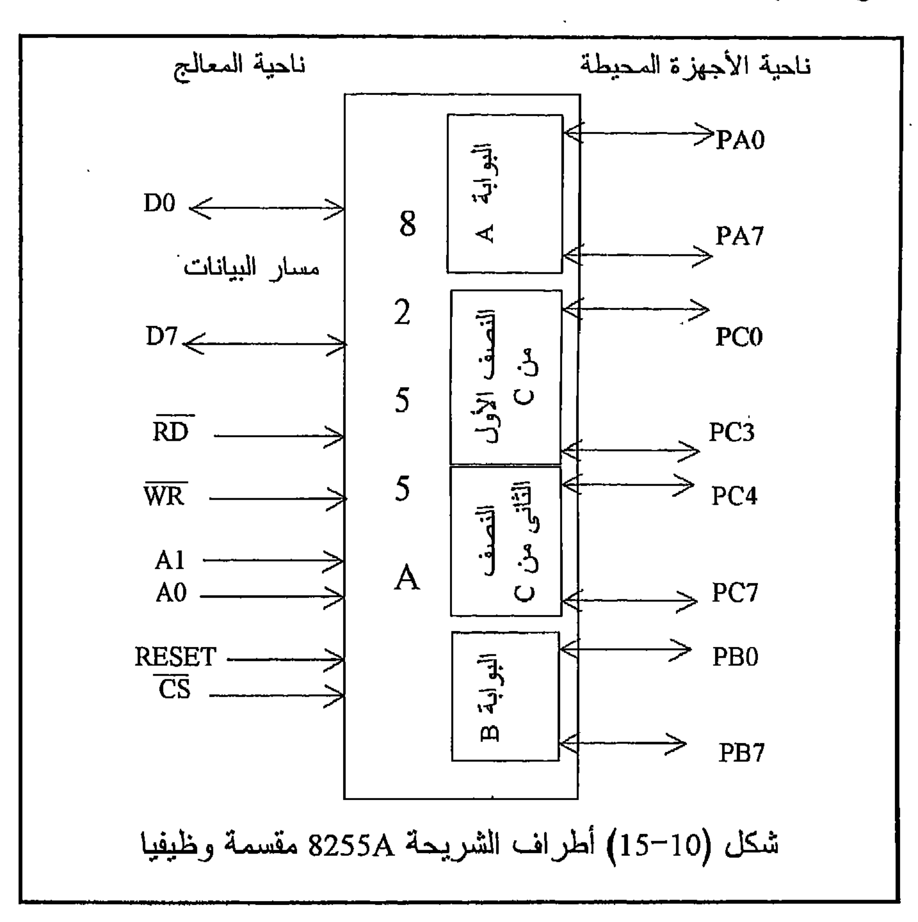
10-5 البوابات القابلة للبرمجة
Programmable Peripheral Interface, PPI

الشريحة Intell 8255A هى إحدى الشرائح المهمة التى تصـــاحب دائمـــا وفـــى الكثير من التطبيقات شرائح المعالج . إن هذه الشريحة تســـمى Programmable Peripheral Interface واختصارا تكتب PPI . تحتوى هذه الشريحة كما ســـنرى على ثلاث بوابات كل منها ثمانى بتات وهذه البوابات قابلة للبرمجة بمعنى أنـــه من داخل البرنامج يمكن جعل أى واحدة من هذه البوابات بوابة إدخال أو إخـــراج عن طريق أمرين فقط من أوامر لغة الأسمبلى وهذه تعتبر من المميزات العظيمة

لهذه الشريحة حيث أنها توفر الكثير من الدوائر التى كانت تستخدم فـــى عمليـــة تشفير البوابات ، فلك أن تتخيل أن أمامك ثلاث بوابات لك الحرية فى أن تجعـــل أى واحدة منهم إدخال أو إخراج كما تشاء ومن داخل البرنامج ودون اللجوء إلى أى تغيرات أو تعديلات فى الدوائر التى تم بناؤها .

10-5-1 تركيب الشريحة 8255A

هذه الشريحة يمكن تقسيمها إلى جزأين رئيسيين ، جزء يواجه شريحة المعـــالج والجزء الآخر يواجه العالم الخارجى . شكل (10-15) يبين رسما تخطيطيا لـــهذه الشريحة محتويا هذين الجزأين . الجزء المقـــابل للميكروبروسيســـور يحتـــوى الخطوط التالية :



شكل (10-15) أطراف الشريحة 8255A مقسمة وظيفيا

- خطوط مسار البيانات وعددها   8 خطوط
- خطوط مسار العناوين وعددها   خطان

- خطوط مسار التحكم وعددها   4 خطوط ($\overline{RD}$, $\overline{WR}$, $\overline{CS}$, reset)

أما الجزء المقابل للعالم الخارجى فيحتوى الخطوط التالية :
- خطوط البوابة A وعددها   8 خطوط
- خطوط البوابة B وعددها   8 خطوط
- خطوط البوابة C وعددها   8 خطوط (4 خطوط + 4 خطوط)

بإضافة خطى القدرة (Vcc والأرضى) يصبح عدد خطوط أو أرجل هذه الشريحة 40 رجلا . لاحظ أن البوابة C لها خاصية منفردة عن البوابتين B, A وذلك لأنها يمكن برمجتها كبوابة 8 بتات أو كبوابتين كل منهما 4 بتات أو أنها تستخدم كخطوط تحكم للبوابتين B, A كما أنه يمكن عمل Set أو Reset لأى طرف من أطراف هذه البوابة بالذات كما سنرى فيما بعد . شكل (10-16) يبين الرسم الطرفى لهذه الشريحة .

8255A			
PA3	1	40	PA4
PA2	2	39	PA5
PA1	3	38	PA6
PA0	4	37	PA7
$\overline{RD}$	5	36	$\overline{WR}$
$\overline{CS}$	6	35	RESET
GND	7	34	D0
A1	8	33	D1
A0	9	32	D2
PC7	10	31	D3
PC6	11	30	D4
PC5	12	29	D5
PC4	13	28	D6
PC0	14	27	D7
PC1	15	26	Vcc
PC2	16	25	PB7
PC3	17	24	PB6
PB0	18	23	PB5
PB1	19	22	PB4
PB2	20	21	PB3

شكل (10-16) أطراف الشريحة 8255A .

من وجهة نظر المعالج فإن الشريحة 8255 تحتوى على أربعة مسجلات أو أربع بوابات أو أربعة أماكن يمكن عنونتها بعنوان خاص لكل منها وهذا هو السبب فى

أن الشريحة 8255 لها خطان فقط للعناوين . ثلاثة من هذه المسجلات أو الأماكن هى المسجلات أو البوابات C, B, A وهى التى تستخدم لإدخال أو إخراج المعلومات وهى الثلاث بوابات التى فى شكل (10–15) ، أما المسجل الرابع فهو مسجل تحكم Control register والذى عن طريقه يتم التحكم فى المسجلات C, B, A لجعلها إما بوابات إدخال أو إخراج والتحكم أيضا فى طريقة تشغيل الشريحة ككل .

العملية	$\overline{CS}$	$\overline{WR}$	$\overline{RD}$	A0	A1
قراءة من البوابة A	0	1	0	0	0
قراءة من البوابة B	0	1	0	1	0
قراءة من البوابة C	0	1	0	0	1
كتابة فى البوابة A	0	0	1	0	0
كتابة فى البوابة B	0	0	1	1	0
كتابة فى البوابة C	0	0	1	0	1
كتابة فى مسجل التحكم	0	0	1	1	1
الشريحة غير فعالة	1	x	x	x	x
حالة غير ممكنة	0	1	0	1	1
الشريحة غير فعالة	0	1	1	x	x

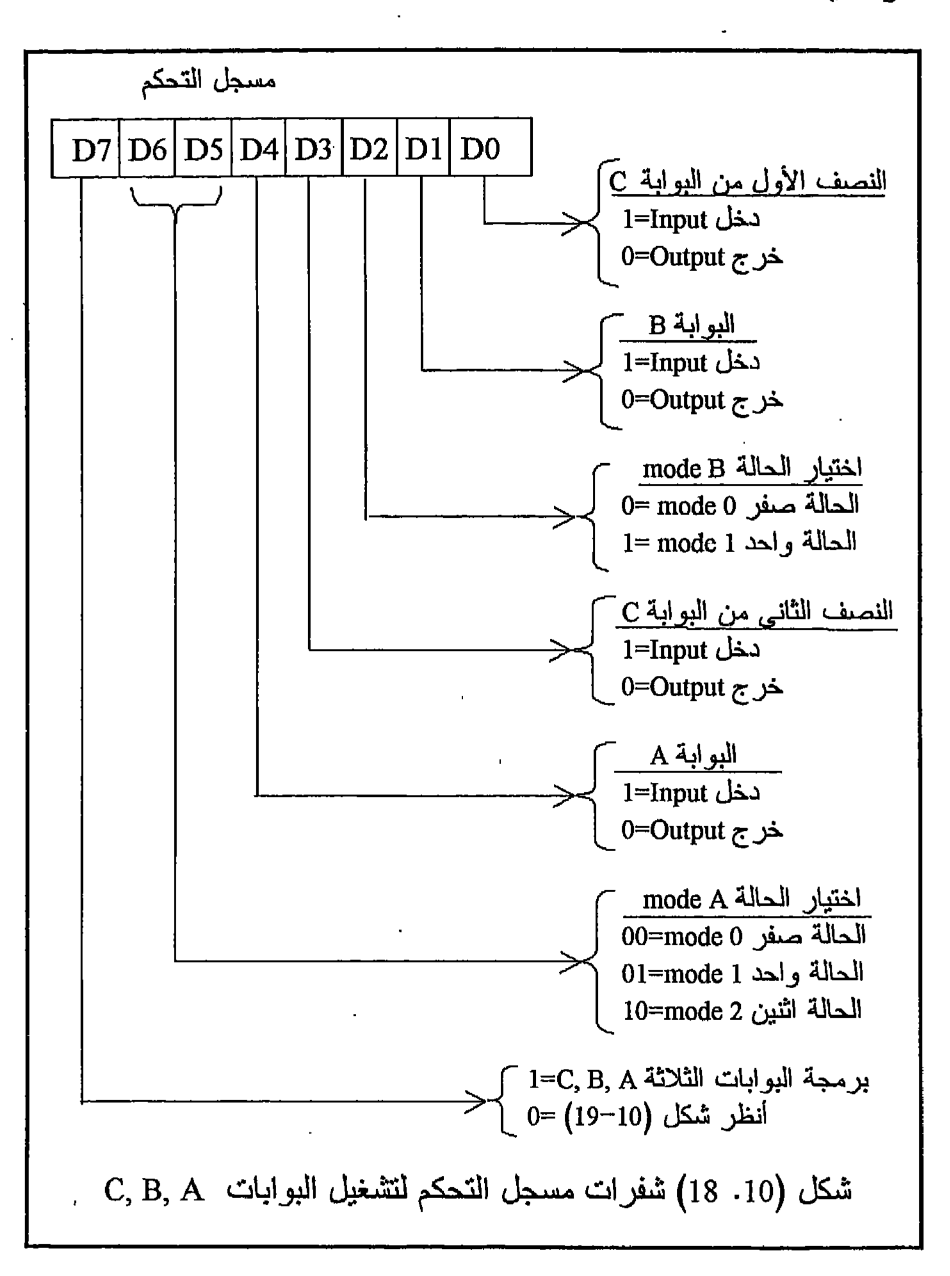
شكل (10–17) عنونة المسجلات فى الشريحة 8255A . (x تعنى لا يهم)

شكل (10–17) يبين كيفية عنونة كل واحد من المسجلات C, B, A وكذلك مسجل التحكم وكيفية القراءة منها أو الكتابة فيها . فمثلا لكى نرسل معلومة إلى المسجل A فإننا نضع الشفرة 00 على الخطين A1, A0 ونجعل خط الكتابة $\overline{WR}$ فعالا بجعله يساوى صفرا وكذلك الخط $\overline{CS}$ وهو خط اختيار الشريحة لابد وأن يكون أيضا فعالا بجعله يساوى صفرا . تتبع طرق القراءة والكتابة فى كل واحد من هذه المسجلات فى شكل (10–17) . لاحظ أن مسجل التحكم يمكن الكتابة فيه فقط ولا يمكن قراءته حيث أن وظيفته لا تتطلب قراءته ، لذلك فإنه يسمى مسجل كتابة فقط write only register .

10–5–2 برمجة الشريحة 8255A

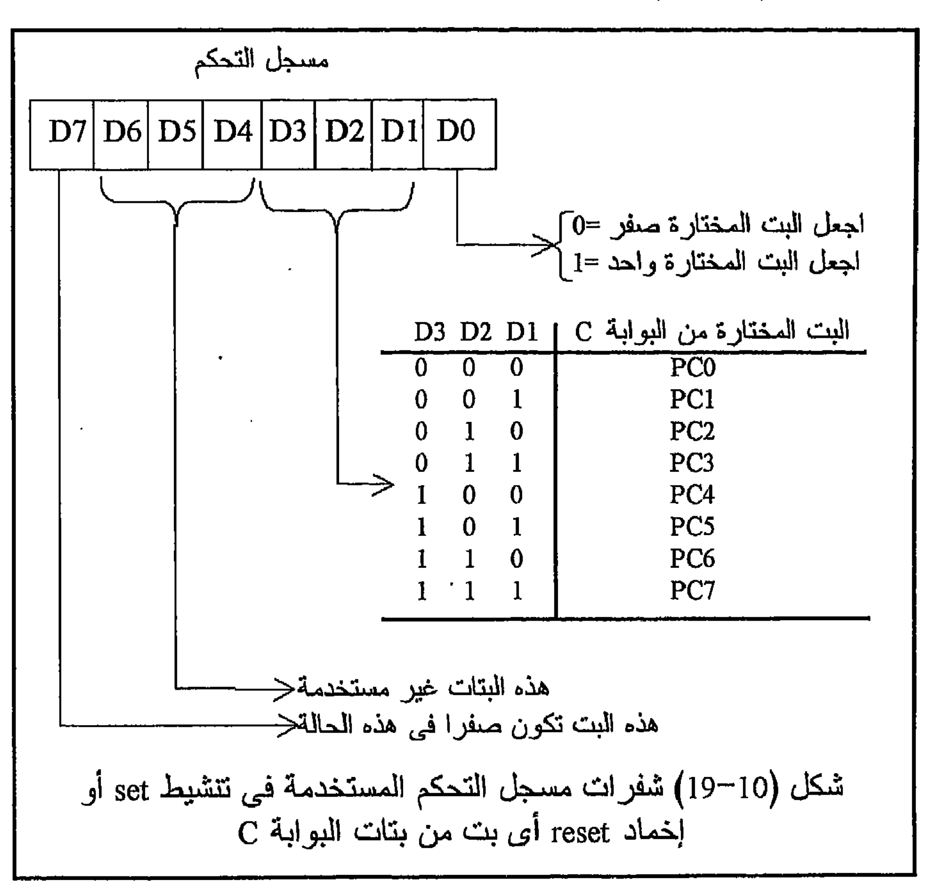
إن برمجة الشريحة 8255A لتعمل فى أى وضع من أوضاع تشغيلها يتم عن طريق كتابة شفرة من ثمانى بتات فى مسجل التحكم ، وعلى ضوء هذه الشفرة تصبح أى واحدة من البوابات الثلاث فى الوظيفة والحالة mode التى حددت لها

تبعا لهذه الشفرة . تذكر هنا أن من الوظائف الخاصة بالبوابة C أنك يمكنـك أن تنشط أو تخمد (set أو reset) أى طرف من أطرافها كما ذكرنا سـابقا . شـكل (10-18) وشكل (10-19) يبينان وظيفة مسجل التحكم علـى ضـوء الشـفرة المخزنة فيه .



شكل (10. 18) شفرات مسجل التحكم لتشغيل البوابات C, B, A .

يهمنا هنا محتويات البت رقم 7 من مسجل التحكم فإذا كانت هذه البت تساوى واحدا فإن ذلك يعنى أن باقى محتويات مسجل التحكم وهى البت رقم صفر إلى البت رقم 6 خاصة بتحديد الوظائف المختلفة للبوابات C, B, A على ضوء ما هو مبين فى شكل (10-18) . أما إذا كانت البت رقم 7 من مسجل التحكم تساوى صفرا فإن ذلك يعنى أن الشفرة الموجودة فى باقى البتات خاصة بعمل set أو reset لأحد أطراف البوابة C على ضوء ما هو مبين فى شكل (10-19) وتسمى حالة تنشيط/إخماد البت Bit Set Reset واختصارا تكتب BSR وسنتعارف عليها بالعربى هكذا (ب س ر) .



D3	D2	D1	البت المختارة من البوابة C
0	0	0	PC0
0	0	1	PC1
0	1	0	PC2
0	1	1	PC3
1	0	0	PC4
1	0	1	PC5
1	1	0	PC6
1	1	1	PC7

شكل (10-19) شفرات مسجل التحكم المستخدمة فى تنشيط set أو إخماد reset أى بت من بتات البوابة C

مثال 10-1

ما هى الشفرة أو البايت التى نكتبها فى مسجل التحكم للشريحة 8255A للحصول على الآتى :

البوابة B دخل ، والبوابة A خرج ، والبوابة C خرج ، والجميع يعمل فى mode 0 . سنفترض الحالة 0 فقط حاليا إلى أن يتم شرح باقى الحالات فى الجزء القادم .

بالنظر إلى شكل (10-18) نجد أنه لكى يكون النصف الأول من البوابة C خرجا فإن D0=0 ولكى تكون البوابة B دخلا فإن D1=1 ولكى تكون المجموعة B فى mode 0 فإن D2=0 ، بالنسبة للنصف الثانى من البوابة C فلكى يكون خرجا فإن D3=0 ولكى تكون البوابة A خرجا فإن D4=0 ولكى تعمل المجموعة A فى mode 0 فإن D6D5=00 ، وكما نعلم فإن D7=1 فى هذه الحالة . بذلك تصبح محتويات البايت المطلوبة هى :

D7 D6 D5 D4 D3 D2 D1 D0
1 0 0 0 0 0 1 0

بالنظام الستعشرى فإن هذه البايت تكون 82H وال H تستخدم للدلالة على أن الرقم المجاور لها يكون فى النظام الستعشرى . المطلوب الآن هو كتابة أو إرسال هذا الرقم إلى مسجل التحكم فى الشريحة 8255A . لإرسال هذا الرقم إلى مسجل التحكم فى الشريحة 8255A فإنه كما نعلم أن خطى العناوين A1, A0 للشريحة 8255A موصلان بخطى العناوين A1, A0 القادمين من شريحة المعالج ولقد رأينا فى شكل (10-17) أن عنوان مسجل التحكم هو A1A0= 11 لذلك فإنه لكى نرسل الرقم المطلوب إلى مسجل التحكم يكفى أن نكتب الأمرين التاليين :

```
MVI A,82H
OUT 03
```

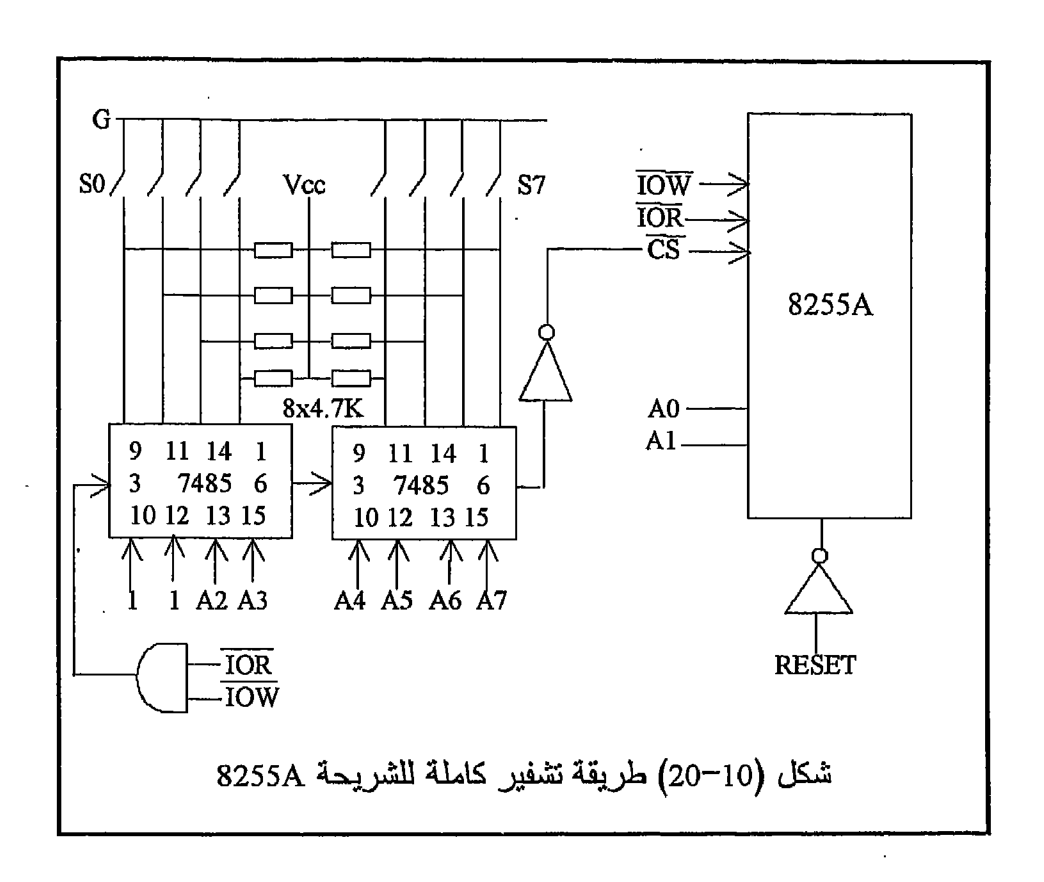
حيث الأمر الأول سيضع الرقم 82H فى مسجل التراكم A والأمر الثانى سيخرج محتويات مسجل التراكم إلى البوابة رقم 3 وهى مسجل التحكم فى الشريحة 8255A لأن الرقم 3 سيجعل خطى العناوين A1, A0 يساويان 11 وهذا هو عنوان مسجل التحكم .

بذلك يمكن التعامل مع البوابات C, B, A حسب حالة كل منهم التى تم تحديدها فى مسجل التحكم بالرقم 82H . فمثلا الأمر OUT 00 سيخرج محتويات مسجل التراكم على البوابة A لأن عنوان البوابة A هو A1A0=00 . وأما الأمر IN 01 فسيقرأ محتويات البوابة B ويضعها فى مسجل التراكم لأن عنوان البوابة B هو A1A0=01 . وأما الأمر OUT 02 فسيقوم بعملية إخراج على البوابة C حيث عنوان البوابة C هو A1A0=10 . لاحظ أن العناوين التى استخدمناها للبوابات الثلاث فى الأوامر السابقة كانت 00 و 01 و 02 و 03 وكلها أرقاما ستعشرية وليس بالضرورة أن تكون هذه هى الأرقام التى يجب أن تستخدم فقط فى جميع الأحوال . النظرية هنا هى أن العنوان الذى نستخدمه يجب أن يحقق الشفرة المطلوبة لكل بوابة على الخطين A1, A0 . شكل (10-20) يبين الشريحة 8255A وقد استخدمت مع عملية تشفير كاملة للثمانية خطوط الأولى من مسار

العناوين . لاحظ عدم استخدام الخطين A1, A0 فى عمليـــة التشـــفير ولكنـــهما يوصلان مباشرة إلى الشريحة 8255A .

من شكل (10−20) نتبين العناوين التالية للمسجلات الأربعة :

العنوان ستعشرى	A7	A6	A5	A4	A3	A2	A1	A0	
0E	0	0	0	0	1	1	0	0	A
0D	0	0	0	0	1	1	0	1	B
0E	0	0	0	0	1	1	1	0	C
0F	0	0	0	0	1	1	1	1	CR مسجل التحكم



شكل (10−20) طريقة تشفير كاملة للشريحة 8255A

10−5−3 حالات Modes تشغيل الشريحة 8255A

الشريحة 8255A لها ثلاث حالات modes لتشغيلها وهى كالتالى :

10-5-3-1 الحالة صفر Mode zero

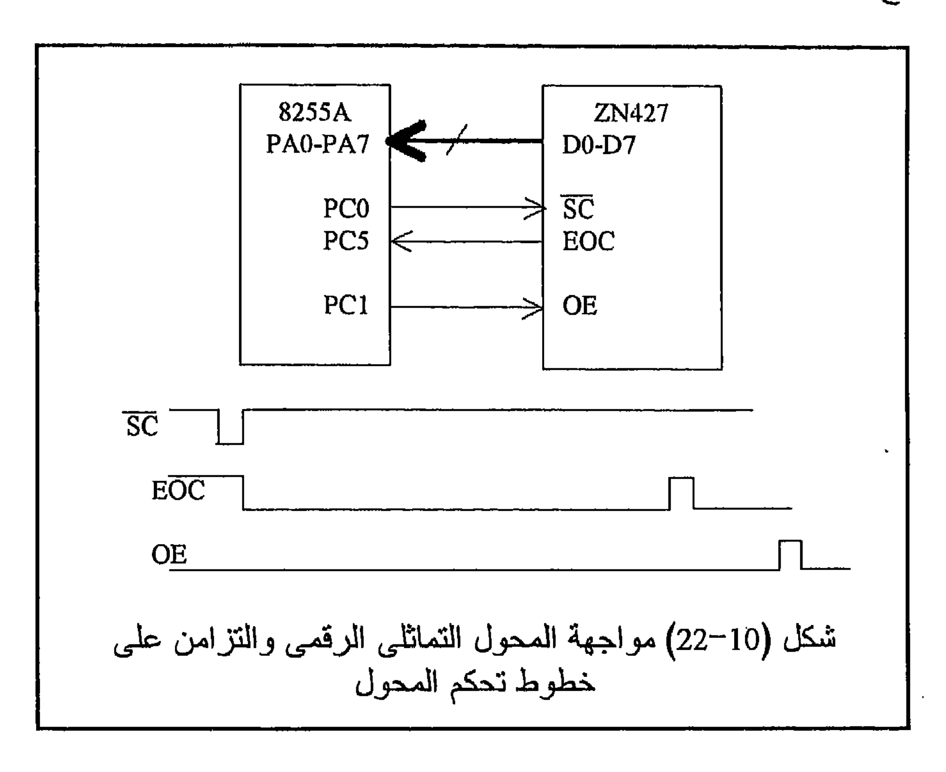
تتميز هذه الحالة بعمليات الإخراج والإدخال البسيطة التى لا تتطلب حـــوارا أو نظام مصافحة handshaking بين الشريحة والمعالج ، ففى هــذه الحالــة يمكــن برمجة كل من البوابة A أو B أو C لتكون بوابة إخراج أو إدخال فى أى مكــان فى البرنامج وعن طريق أمرين اثنين فقط كما سنرى بعد قليل . لاحظ أنه فـــى هذه الحالة يمكن برمجة البوابة C كبوابة ثمانى بتات أو بوابتين كل منهما أربــع بتات . جميع هذه البوابات تكون فيها خاصية المسك latch للمعلومة عندما تعمـل كبوابات إخراج ، وأما عندما تعمل كبوابات إدخال فلا تكون فيها هذه الخاصية . باعتبار البوابة C كبوابتين كل منهما 4 بتات فإنه يمكن أن يكون لدينا 4 بوابــات يمكن برمجتها إخراج أو إدخال على حسب الشفرة التى ترسل إلى مسجل التحكم على ضوء ما رأينا فى شكل (10-18) . شكل (10-21) يبين جميع هذه الحالات والشفرة الستعشرية التى ترسل إلى مسجل التحكم للحصول على كل حالة .

الشفرة الستعشرية إلى مسجل التحكم	البوابة A	البوابة C PC4-PC7	البوابة B	البوابة C PC0-PC3
80	إخراج	إخراج	إخراج	إخراج
81	إخراج	إخراج	إخراج	إدخال
82	إخراج	إخراج	إدخال	إخراج
83	إخراج	إخراج	إدخال	إدخال
88	إخراج	إدخال	إخراج	إخراج
89	إخراج	إدخال	إخراج	إدخال
8A	إخراج	إدخال	إدخال	إخراج
8B	إخراج	إدخال	إدخال	إدخال
90	إدخال	إخراج	إخراج	إخراج
91	إدخال	إخراج	إخراج	إدخال
92	إدخال	إخراج	إدخال	إخراج
93	إدخال	إخراج	إدخال	إدخال
98	إدخال	إدخال	إخراج	إخراج
99	إدخال	إدخال	إخراج	إدخال
9A	إدخال	إدخال	إدخال	إخراج
9B	إدخال	إدخال	إدخال	إدخال

شكل (10-21) جميع حالات الإدخال والإخراج للبوابات A و B و C والشفرة الستعشرية لكل حالة .

مثال 10-2

شكل (10-22) يبين رسما صندوقيا لدائرة مواجهة مع المحول التماثلي الرقمي ZN427 مستخدما الشريحة 8255A في الحالة 0 . المحول ZN427 لـه ثلاثة خطوط تحكم ، أحدها هو خط بداية التحويل $\overline{SC}$ Start Conversion; وهذا الخط يجب أن يكون صفرا لفترة زمنية وجيزة بحيث يبدأ المحول عملية التحويل عند الحافة الصاعدة لهذه الإشارة . بعد أن ينتهي المحول من عملية التحويل فإنه يعطي نبضة واحد على خط نهاية التحويل End Of Conversion; EOC ولكنه لا يعطي البيانات على خطوط الخرج الثمانية إلا إذا تم إعطاؤه نبضة واحد على خط تنشيط الخرج Output Enable; OE الذي يقوم بتنشيط البوابات الثلاثية الموجودة في خرج المحول . شكل (10-22) يبين أيضا التزامن الموجود بين هذه الإشارات الثلاثة . لقد تم توصيل خرج المحول على البوابة A للشريحة 8255A ، أي أن البوابة A ستعمل كبوابة إدخال يقوم المعالج من خلالها بقراءة خرج المحول .



شكل (10-22) مواجهة المحول التماثلي الرقمي والتزامن على خطوط تحكم المحول

مطلوب أيضا من المعالج أن يعطي للمحول نبضة بدأ التحويل $\overline{SC}$ وسيكون ذلك من خلال الخط رقم 0 في البوابة C وسيكون عن طريق استخدام الحالة (ب س ر) التي سنستخدمها لتنشيط الخط PC0 كما رأينا في شكل (10-19) . بعد أن

يعطى المعالج نبضة بداية التحويل يبدأ فى مراقبة خط نهاية التحويل القادم من المحول إلى الخط PC5 بحيث عندما يجد المعالج أن هذا الخط صعد إلى الواحد يفهم من ذلك أن المحول انتهى من عملية التحويل فيعطى له إشارة تنشيط الخرج بجعل الخط OE الموصل على PC1 يساوى واحد وذلك أيضا باستخدام طريقة ال (ب س ر) . لذلك فإن النصف الأول من البوابة C سيعمل كبوابة إخراج وأما النصف الثانى فسيعمل كبوابة إدخال . البرنامج التالى يمكن استخدامه لقراءة خرج مثل هذا المحول . ملاحظة مهمة يجب أن نتذكرها من هذا البرنامج وهى أن إرسال أى شفرة إلى مسجل التحكم لتنشيط أو إخماد أى بت من بتات البوابة C لا يؤثر على الإطلاق على الحالة التى عليها كل من البوابة A أو B أو C ، أى أن كل واحدة منها تبقى على الحالة التى عليها سواء كانت إدخالا فستبقى إدخالا أو كانت إخراجا فستبقى كذلك أيضا .

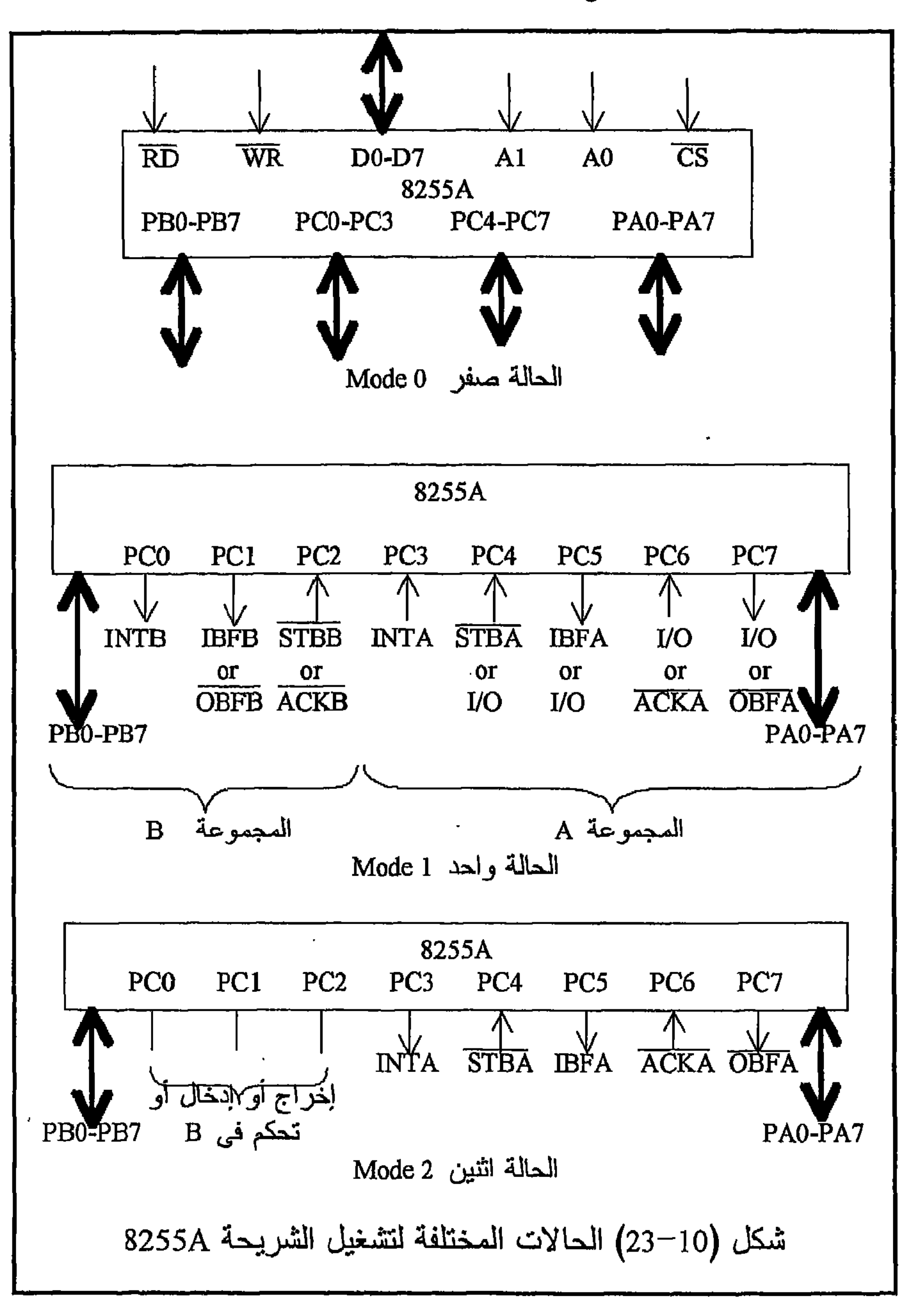
```
MVI A,98H ;      هذا الرقم يجعل PC0-PC3 إخراج والبوابة B إخراج
;                (B غير مستخدمة فى هذا المثال) و PC4-PC7 إدخال والبوابة A إدخال
;                راجع شكل (10-18).
OUT 03H   ;      إخراج الرقم 98H إلى مسجل التحكم
START: MVI A,01
;                تنشيط الخط PC0 عن طريق ب س ر راجع شكل (10. 19)
OUT 03H   ;      إخراج إلى مسجل التحكم
MVI A,00  ;      عمل إخماد للخط PC0
OUT 03H   ;      إخراج إلى مسجل التحكم
MVI A,01  ;      عمل ست للخط PC0
OUT 03H   ; SC.  بذلك يكون الخط نزل من 1 إلى 0 ثم صعد وبذلك تتم نبضة
xx: IN 10 ;      قراءة البوابة C
ANA 20H   ;      هذا لاختبار الخط PC5 لمعرفة إذا كان واحد أم صفر
 JZ xx    ;      قفز إلى xx لمعاودة القراءة طالما أن PC5 يساوى 0 .
MVI A,03H ;      تنشيط للخط PC1 لتنشيط الخط OE .
OUT 03H   ;      إخراج إلى مسجل التحكم
MVI A,02H ;      إرجاع الخط OE=PC1 إلى الصفر ثانية.
OUT 03H   ;      إخراج إلى مسجل التحكم
IN 00     ;      قراءة البوابة A
JMP START ;      لإعطاء نبضة بدأ تحويل جديدة
```

10-5-3-2 الحالة واحد Mode one

فى هذه الحالة يتم الإدخال أو الإخراج عن طريق البوابتين A و B فقط ويتم ذلك عن طريق نظام مصافحة Handshaking أو حوار بين الأجهزة المحيطة والمعالج من خلال الشريحة 8255A وتستخدم خطوط البوابة C فى عمليات المصافحة هذه بحيث أن النصف الأول من خطوط البوابة C وهى PC0 إلى PC3 تستخدم كخطوط حوار تابعة للبوابة B ولذلك تسمى البوابة B والنصف الأول من البوابة C بالمجموعة B أو Group B . وأما النصف الثانى من خطوط البوابة C فيستخدم كخطوط مصافحة أو حوار تابعة للبوابة A ولذلك تسمى البوابة A والخطوط PC4 إلى PC7 بالمجموعة A . لاحظ أن البوابة C بأكملها لا يمكن استخدامها هنا فى الإخراج أو الإدخال ولكن يمكن استخدام بعض خطوطها أحيانا كما فى شكل (10-23) . تتميز هذه الحالة بأن البيانات سواء الداخلة أو الخارجة من البوابتين A أو B تمسك latched على هذه البوابات ، كما أن هذه الحالة توفر للمستخدم خطا يمكن منه مقاطعة المعالج . عمليات المقاطعة سندرسها بالتفصيل فى فصل خاص بذلك . شكل (10-23) يبين وظيفة كل خط من خطوط البوابة C كخطوط مصافحة لكل من البوابتين A و B فى الحالة 1 وذلك عندما تستخدم كل من A و B كبوابة إدخال أو إخراج . شكل (10-24) يبين وظيفة خطوط البوابة C فى حالة كون كل من البوابتين A و B بوابات إدخال فقط ويبين أيضا التزامن بين هذه الإشارات فى هذه الحالة . لاحظ أن البوابة B تستخدم الخطوط PC0, PC1, PC2 كخطوط مصافحة بينما تستخدم البوابة A الخطوط PC3, PC4, PC5 كخطوط مصافحة ويتبقى خطان وهما PC6, PC7 يستخدمان كخطوط إدخال أو إخراج على حسب البت D3 فى مسجل التحكم . تتم عملية المصافحة مع الأجهزة المحيطة كما يلى :

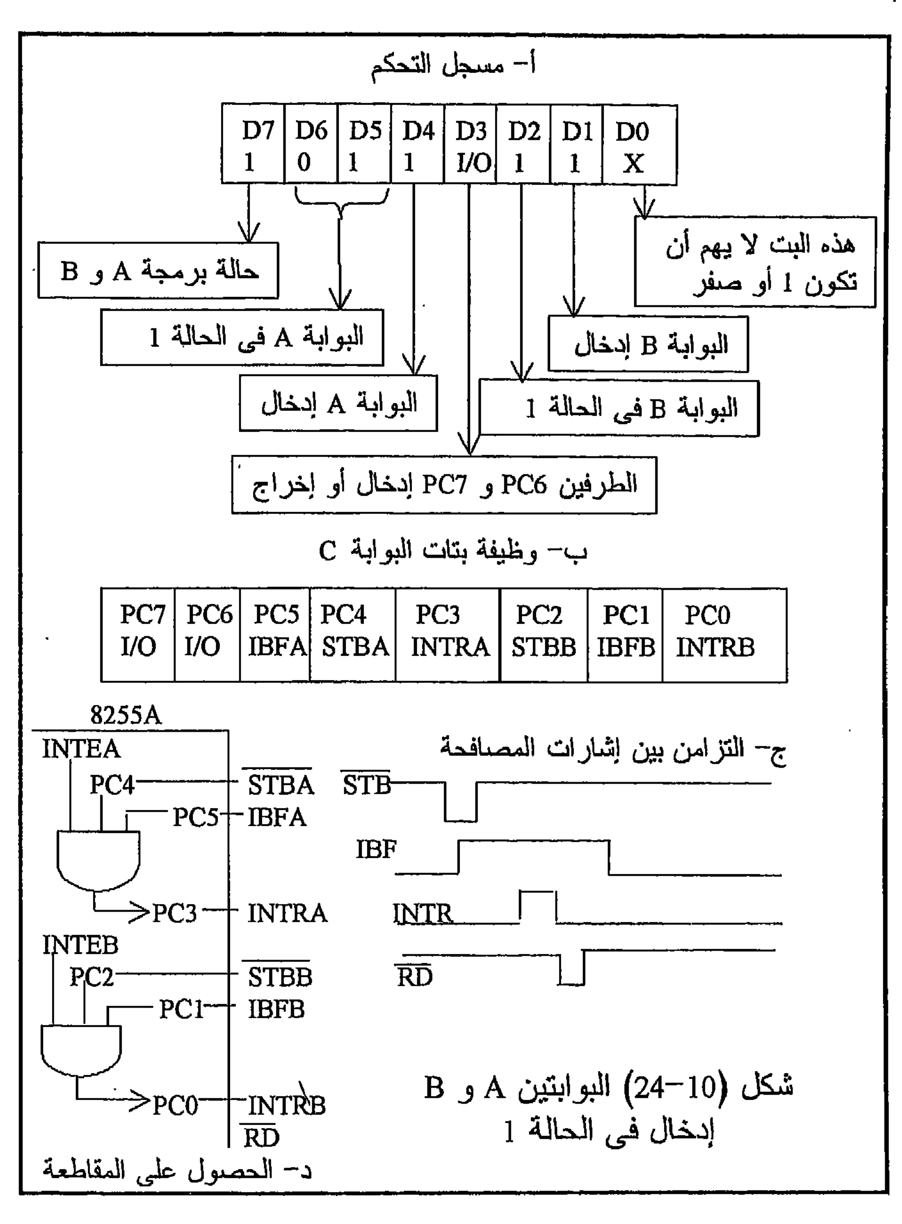
- يقوم الجهاز الخارجى بوضع البيانات على البوابة التى يتعامل معها ولتكن البوابة A مثلا ثم يضع صفرا على الخط $\overline{STBA}$ ليخبر الشريحة 8255A أنه قد وضع بايت على البوابة A .
- ترد الشريحة 8255A على الجهاز الخارجى بأنها استقبلت المعلومة وتم مسكها على البوابة A عن طريق جعل الخط IBFA يساوى واحدا . عندما يرى الجهاز الخارجى أن الخط IBFA=1 يقوم بإرجاع الخط $\overline{STBA}$ إلى الواحد ثانية استعدادا لإرسال بايت أخرى إن أراد . لاحظ أن الخط IBFA لا ينزل إلى الصفر ثانية إلا إذا تم قراءة المعلومة بواسطة المعالج وذلك عند الحافة الصاعدة للإشارة $\overline{RD}$ الموصلة بالشريحة 8255A طرف 5.
- تقوم الشريحة 8255A بتوليد إشارة مقاطعة INTRA يمكن بها مقاطعة المعالج فى حالة الرغبة فى ذلك وذلك بتوصيلها إلى أى خط من خطوط المقاطعة على المعالج . لاحظ أيضا من التزامن فى شكل (10-24) أن هذا

الخط يكون واحدا إذا كان الخط STBA=1 والخط IBF=1 والبت PC4=1 فـــى حالة التعامل مع البوابة A وأما فى حالة البوابـــة B فـــالبت PC2=1 . البتـــات PC4, PC2 يمكن جعلها واحدا باستخدام ال ب س ر كما ذكرنا سابقا وتذكـــر أن ذلك ليس له تأثير على وضع البوابات .



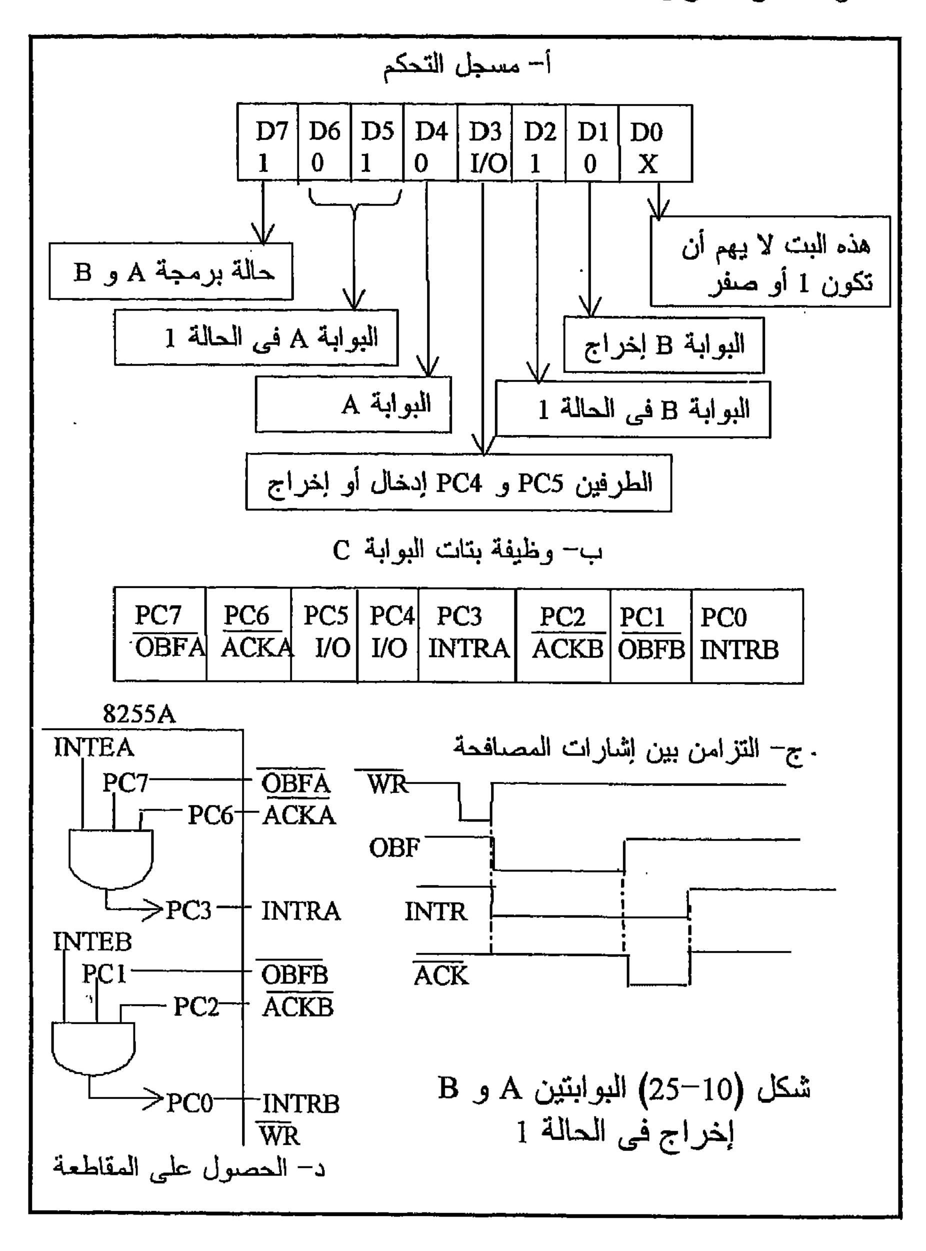
شكل (10-23) الحالات المختلفة لتشغيل الشريحة 8255A

شكل (10−25) يبين نفس الوظائف لخطوط البوابة C ولكن فى حالة كـون A و B بوابات إخراج ويبين أيضا التزامن بين هذه الإشارات . لاحظ أن البوابـة B تستخدم الخطوط PC0, PC1, PC2 كخطوط مصافحة بينما تسـتخدم البوابـة A الخطوط PC3, PC6, PC7 كخطوط مصافحة ويتبقى خطان وهمـا PC4, PC5 يستخدمان كخطوط إدخال أو إخراج على حسب البت D3 فى مسجل التحكـم .
تتم عملية المصافحة كما يلى :



شكل (10−24) البوابتين A و B إدخال فى الحالة 1

• عندما يقوم المعالج بكتابة أى بايت إلى أى من البوابتين A أو B فإنه عند الحافة الصاعدة للإشارة $\overline{WR}$ تقوم الشريحة 8255A بجعل الخط $\overline{OBFA}$ يساوى صفرا دلالة للمعالج أن هذه المعلومة قد تم مسكها كما فى التزامن الموضح فى شكل (10-25) وعلى المعالج ألا يرسل معلومات أخرى إلا بعد صعود هذا الخط للواحد مرة أخرى .



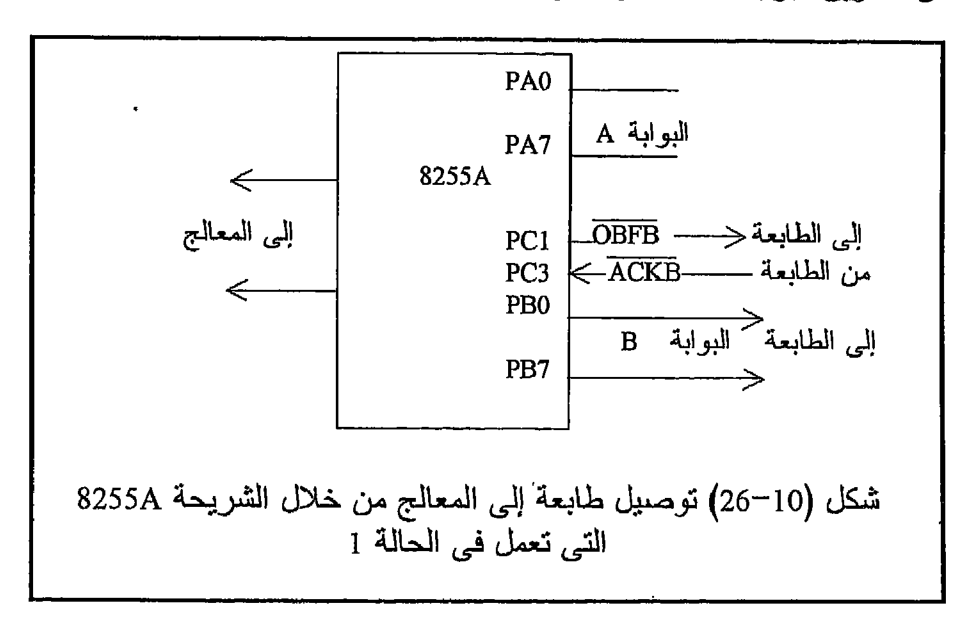
شكل (10-25) البوابتين A و B إخراج فى الحالة 1

• عندما يرى الجهاز الخارجى أن الخط $\overline{OBF}$ يساوى صفرا يعرف أن هنـــاك بايت على البوابة وعليه قراءتها فيقوم بجعـــل الخـــط $\overline{ACK}$ يســـاوى صفـــرا وإرجاعه مرة أخرى للواحد لكى يخبر الشريحة 8255A أنه قد قرأ المعلومـــة . لاحظ من التزامن أن الحافة النازلة للخط $\overline{ACK}$ تتسبب فى إرجاع الخـــط $\overline{OBF}$ إلى الواحد مرة أخرى .

• تقوم الشريحة 8255A بتوليد إشارة مقاطعة INTRA يمكـــن بـــها مقاطعـــة المعالج فى حالة الرغبة فى ذلك وذلك بتوصيلها إلـــى أى خـــط مـــن خطـــوط المقاطعة على المعالج بغرض طلب بايت جديدة منه . لاحظ أيضا من الـــتزامن فى شكل (10-25) أن هذا الخط ينزل إلى الصفر مع الحافة الصاعدة للخط $\overline{WR}$ ويرجع إلى الواحد مرة ثانية عندما يكـــون كـــل مـــن الخطيـــن $\overline{ACK}$ و $\overline{OBF}$ يساوى واحدا والبت PC6 فى حالة التعامل مع البوابة A والبت PC2 فى حالـــة البوابة B تساوى واحدا أيضا . البتات PC6, PC2 يمكن جعلها واحد باســـتخدام ال ب س ر كما ذكرنا سابقا وتذكر أن ذلك ليس له تأثير على وضع البوابات .

مثال 10-3

شكل (10-26) يبين عملية توصيل طابعة مع المعالج مـــن خـــلال البوابـــة B للشريحة 8255A وشكل (10-27) يبين البرنامج الذى يقرأ محتويـــات الذاكـــرة ابتداء من المكان E100H وانتهاء بالمكان E150H ويرسلها إلى الطابعة للطباعة. إن عملية تشفير البوابات الثلاث ، أى عنونتها سنتركها للقارئ يختار مـــا يشـــاء من عناوين للبوابات الثلاثة وسنفترض هنا أن هذه البوابات معنونة كالتالى :



شكل (10-26) توصيل طابعة إلى المعالج من خلال الشريحة 8255A التى تعمل فى الحالة 1

00H البوابة A
01H البوابة B
02H البوابة C
03H مسجل التحكم

فى هذا المثال تم استخدام البوابة B فقط وفى الحالة mode واحد وأما البوابة A وباقى البوابة C فيمكن استخدامها لأى أغراض أخرى . السؤال هو : ما هى الشفرة التى سنرسلها إلى مسجل التحكم لنجعل البوابة B إخراج وفى الحالة 1 وأما البوابة A وباقى البوابة C فسنفترضها إخراج فى الحالة 0 وذلك مع العلم أنها لن تستخدم ؟ الإجابة هى أن هذهِ الشفرة ستكون كالتالى كما تعلمنا من شكل (10-18) :

D7 D6 D5 D4 D3 D2 D1 D0
1 0 0 0 0 1 0 0 = 84H

- النصف الأول من البوابة C لا يهم أن يكون إخراجا أو إدخالا وبفرضه إخراجا تم وضع D0=0 .
- بما أن البوابة B ستكون إخراجا لذلك تم وضع D1=0 .
- بما أن B ستعمل فى الحالة 1 لذلك تم وضع D2=1 .
- النصف الثانى من البوابة C لا يهم أن يكون إدخالا أو إخراجا وبفرضه إخراجا تم وضع D3=0 .
- بما أن البوابة A فرضت إخراجا لذلك فإن D4=0 .
- بما أن البوابة A فرضت فى الحالة 0 لذلك كان D6D5 = 00 .
- بما أن هذه الشفرة تستخدم لتوظيف البوابات فإن D7=1 .

بالنظر إلى شكل (10-27) نجد أن البرنامج يدأ بإرسال الشفرة 84H كما سبق إلى مسجل التحكم ثم استخدم المسجلين HL كمؤشر إلى الحرف الجاهز للطباعة والمسجل B كعداد تنازلى للأحرف التى سيتم طباعتها بحيث تقف عملية الطباعة عندما يصل هذا العداد إلى الصفر لأن البرنامج ينقص هذا العداد بمقدار واحد كلما تمت طباعة حرف . تبدأ حلقة الطباعة من العلامة NEXT والتى عندها يتم قراءة البوابة C وحجب جميع بتاتها بالقيمة 02H لمعرفة إذا كان الخط $\overline{OBF}$ يساوى صفرا أم لا ، لأن صفرا على هذا الخط معناه أن هناك حرف مازال ممسوكا على البوابة فى انتظار الطابعة لاستلامه . بمجرد أن يصبح الخط $\overline{OBF}$=1 أى غير فعال يخرج المعالج من هذه الحلقة ويقوم بإرسال حرف جديد إلى الطابعة والدخول فى نفس حلقة قراءة البوابة C مرة أخرى . تتكرر هذه العملية إلى أن ينتهى المعالج من جميع الأحرف المطلوب طباعتها عندما يصل المسجل B إلى الصفر .

10-5-3-3 الحالة اثنان Mode two

البوابة A فقط هى التى يمكنها أن تستخدم فى هذه الحالة وأما البوابة B فتكون إما فى الحالة صفر mode 0 أو فى الحالة واحد mode 1 . عندما تشتغل البوابة A فى هذا الحالة فإنها تسلك مسلك مسار بيانات بمعنى أنها تكون بوابة إخراج إذا كانت المعلومات خارجة وتكون بوابة إدخال إذا كانت المعلومات داخلة وذلك دون استخدام أى أمر أو اللجوء إلى مسجل التحكم لعمل ذلك . لذلك فإن البوابة A عندما تكون فى هذه الحالة فإنها تستخدم خمسا من خطوط البوابة C فى نظام المصافحة أو الحوار وهذه الخطوط هى PC3 إلى PC7 . إن هذه الحالة هى أعقد الحالات التى تستخدم مع الشريحة 8255A وعادة يستخدم فى حالات الاتصال بين حاسبين أو بين المعالج والأقراص الصلبة ولذلك سنكتفى بهذه الإشارة عن هذه الحالة .

```
MVI A,84H  ;  البوابة B إخراج فى الحالة 1 و A و C إخراج فى الحالة 0
OUT 03H    ;  إرسال إلى مسجل التحكم
LXI H,E100 ;  إشارة لبداية الأحرف المطلوب طباعتها
MVI B,50H  ;  المسجل B عداد لهذه الأحرف
NEXT: IN 02 ; قراءة البوابة C لمعرفة حالة الخط OBF وهو الخط PC1
ANI 02H    ;  حجب لجميع البتات ما عدا PC1
JZ NEXT    ;  دوران فى الحلقة طالما أن PC1=OBF=0
MOV A,M    ;  إحضار حرف ووضعه فى المركم
OUT 01H    ;  إخراج على البوابة B (الطابعة)
INX H      ;  إشارة إلى الحرف التالى
DCR B      ;  إنقاص العداد بمقدار 1
JNZ NEXT   ;  قفز لطباعة الحرف التالى
```

شكل (10-27) برنامج الطابعة الموصلة فى شكل (10-26)

10-6 تمارين

1. ما المقصود بالإدخال والإخراج ؟
2. ما هو الفرق بين الإدخال والإخراج والكتابة والقراءة من الذاكرة ؟
3. أيهما أسرع ، إرسال واستقبال البيانات على التوالى , أم على التوازى ؟ اذكر بعض التطبيقات التى تستخدم كل نوع ؟

4. عند تنفيذ المعالج للأمر OUT 00 مثلا ، فإنه يقوم بالتأثير علـــى المســـارات الثلاثة ، اذكر هذه التأثيرات ؟
5. عند تنفيذ المعالج للأمر IN 00 مثلا ، فإنه يقوم بالتأثير على المسارات الثلاثة ، اذكر هذه التأثيرات ؟
6. أعد رسم شكلى (10–7 و 10–8) مستخدما فاكك شفرة decoder بـــدلا مـــن مقارن العناوين ؟
7. أعد رسم شكلى (10–9 و 10–10) مستخدما فاكك شفرة decoder بدلا مـــن مقارن العناوين ؟
8. مطلوب توصيل بوابة إخراج واحدة فقط وأخرى إدخال على المعالج ، ارسـم أبسط دائرة للتوصيل ؟
9. ما هو المقصود بالتشفير الكامل والتشفير الناقص ؟ من أى أنـــواع التشـفير تكون الدائرة التى وصلتها فى السؤال الثامن ؟
10. عند تنفيذ المعالج للأمر STA adr مثلا ، يقوم بالتأثير على المسارات الثلاثة ، اذكر هذه التأثيرات ؟ كيف يمكن استغلال هذه التأثيرات لبناء بوابة إخراج ؟
11. عند تنفيذ المعالج للأمر LDA adr مثلا فإنه يقوم بالتأثير علـــى المســـارات الثلاثة ، اذكر هذه التأثيرات ؟ كيف يمكن استغلال هذه التأثيرات لبنـــاء بوابـــة إدخال ؟
12. ماذا تعنى خريطة الذاكرة لأى ميكروكومبيوتر ؟ ارســـم خريطـــة الذاكـــرة للميكروكومبيوتر الذى تستخدمه وادرسها جيدا ؟
13. أعد رسم شكلى (10–12 و 10–13) مستخدما مقارن عناوين بدلا من فـــاكك الشفرة decoder ؟
14. هل يمكن بناء بوابة إخراج تأخذ نفس عنوان أحد أماكن الذاكـــرة الموجـــودة فعلا ؟
15. هل يمكن بناء بوابة إدخال تأخذ نفس عنوان أحد أماكن الذاكـــرة الموجـــودة فعلا؟
16. من وجهة نظر المعالج فإن الشريحة 8255A تتكون من 4 مســـجلات يمكـــن القراءة منها والكتابة فيها ، هل هذه العبارة صح أم خطأ ؟
17. هل من الضرورى أن تكون عناوين البوابات A و B و C متتابعة أى 5A و 5B و 5C مثلا وذلك فى الشريحة 8255A ؟
18. هل يمكن استخدام الشريحة 8255A وعنونتها بطريقة خرائط الذاكرة ؟ أم أنـــه لابد من استخدامها مع الأمرين IN و OUT ؟
19. اشرح مع التبسيط الحالات modes الثلاثة لتشغيل الشريحة 8255A ؟

الفصل الحادى عشر

دراسة لباقى أطراف المعالج من خلال

تطبيق: التحكم في إشارة مرور

Control of a Traffic Light

11-1 مقدمة

إنه فى الكثير من الأحيان وعند ذكر التطبيقات التى تستخدم الميكروكومبيوتر للتحكم فى أى عملية صناعية يتبادر إلى ذهننا فورا الميكروكومبيوتر بصورته المركبة والمعقدة حيث الشاشة ولوحة المفاتيح والطابعة ووحدة التحكم المركزى cpu والكمية الهائلة من الذاكرة التى قد تصل إلى الكثير من الميجابايتات ، فى حين أن العملية من الممكن أن تكون أبسط من ذلك بكثير كما سنرى . سنحاول فى هذا الفصل أن نقدم عرضا مفصلا وشرحا دقيقا لدائرة المعالج على كارت اليكترونى واحد one board وهذا الكارت يمكن استخدامه فى الكثير من أغراض التحكم ومنها على سبيل المثال عملية التحكم فى متغيرات تنك تقطير المياه التى ذكرناها فى مقدمة الفصل العاشر وسوف نستخدم هذه الدائرة فى هذا الفصل فى عملية التحكم فى إشارة مرور رباعية كتطبيق على ذلك .

11-2 تركيب الدائرة

إن الميكروكومبيوتر بمعناه العام والشامل وكما ذكرنا فى الفصل الأول من هذا الكتاب يتركب من شاشة للعرض ولوحة مفاتيح ووحدة تحكم مركزى cpu وكمية من الذاكرة تقل أو تكثر على حسب متغيرات وأوضاع كثيرة . إن الكثير من التطبيقات وبالذات التى تستخدم المعالج فى أغراض التحكم لا تحتاج إلى كل هذه الإمكانيات ، فما فائدة شاشة العرض مثلا فى دائرة نريد تصميمها للتحكم فى سرعة محرك والحفاظ على ثباتها ؟ أو ما فائدة لوحة المفاتيح أو الكمية الهائلة من ال RAM فى دائرة نريد تصميمها للتحكم مثلا فى إشارة مرور فى تقاطع معين داخل مدينة ؟ من هنا كان السؤال عن ما هى أقل الإمكانيات التى نستطيع بها أن نصمم دائرة تكون سهلة البناء ، رخيصة التكاليف ، وتفى بمعظم التطبيقات التى تحتاج إلى المعالج كعنصر أساسى فى تطبيقات التحكم الآلى ؟

تتكون أى دائرة تستخدم المعالج فى أغراض التحكم العامة من الأجزاء التالية :

1. المعالج وقد تم تهيئة جميع مساراته لعملية المواجهة مع الأجهزة المحيطة
2. شريحة ذاكرة EPROM تحتوى البرنامج الذى سيقوم بالعملية التى تستخدم من أجلها دائرة المعالج .
3. شريحة RAM قد تكون هناك الحاجة إليها من قبل البرنامج السابق ، وإذا لم تكن هناك حاجة إليها يمكن فى هذه الحالة الإستغناء عنها .
4. عدد من بوابات الإدخال والإخراج على حسب الحاجة والتطبيق الذى تستخدم من أجله الدائرة المذكورة .

لكى نفهم طريقة عمل هذه الدائرة فنحن نعلم أن المعالج عنـــد إعطائـــه نبضـــة $\overline{RESET}$ أو عند بداية تشغيلة يبدأ التنفيذ من عنوان معين وهذا العنوان يختلـــف من معالج لآخر ، فإذا جعلنا هذا العنوان هو أول عنوان فى البرنامج الذى كتبنـــاه لهذا الغرض (غرض التحكم فى أى عملية صناعية) والموجود فـــى شـــريحة ال EPROM والتى تم توصيلها بحيث تعمل مع هذا العنوان ، فإنه بمجـــرد تشـــغيل المعالج سينفذ البرنامج ويتعامل مع بوابات الإدخال والإخـــراج وعلـــى حســـب ظروف العملية التى صمم من أجلها . المفروض طبعا أن هذا البرنامج ســـيكون برنامجا يدور فى حلقة لا نهائية تجعل المعالج فى حالة تنفيذ مستمرة للبرنامج .

11-2-1 مثال توضيحى

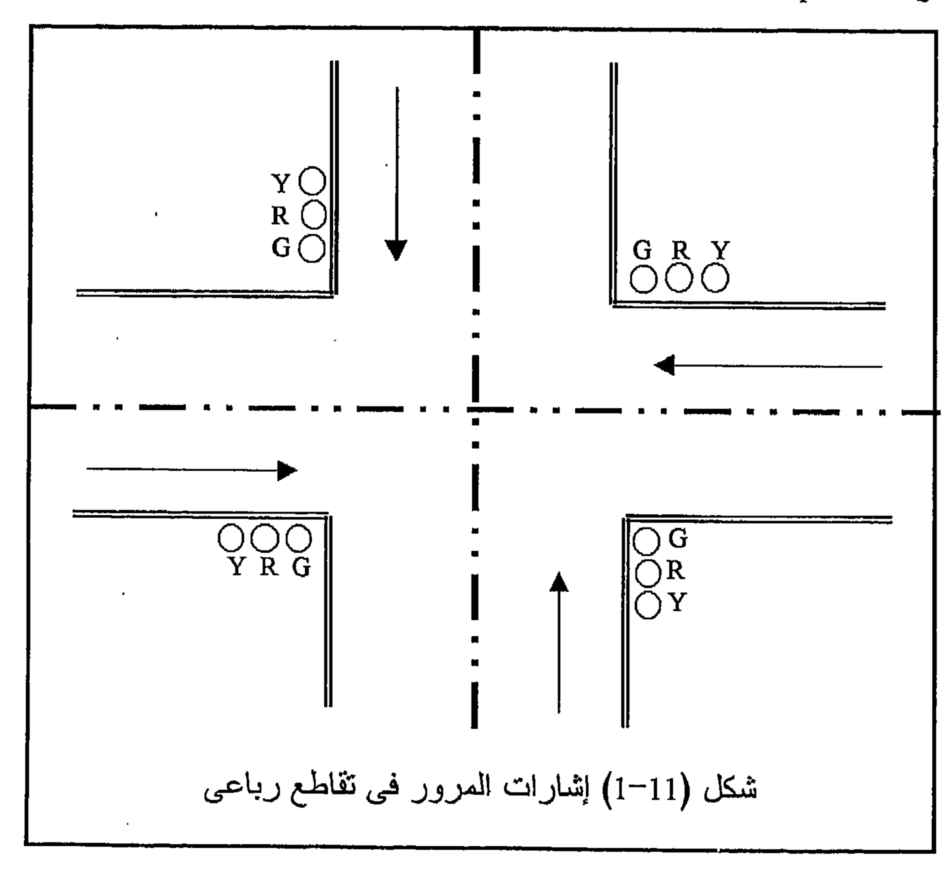
كمثال على ذلك سنقوم فى هذا الفصل إن شاء الله بعمل نظام تحكم فـــى إشـــارة مرور فى تقاطع رباعى كالمبين فى شكل (11-1) بحيث يحتوى كـــرت التحكـــم على مفتاح (خط تحكم يدوى يستعمل بواسطة رجل المرور عند اللزوم) ، بحيـــث عندما يكون هذا المفتاح واحدا فإن الإشارات تعمل فى الوضع الطبيعى وعندمـــا يكون هذا المفتاح صفرا فإن اللون الأصفر فى جميع الإتجاهات يضىء ويطفـــىء flashing بتردد معين وليكن نصف ثانية .

سنقوم هنا ببناء الدائرة مستخدمين المعالج 8085 بعد تجربتها والتأكد من صحتها معمليا ، وسنترك للقارىء حرية إعادة بناء الدائرة مستخدما إما المعـــالج Z80 أو أى معالج آخر إذا كان يفضل العمل بأحدها وسوف نشـــير إلـــى أى معلومـــات ضرورية لذلك فى حينها .

سيقوم المعالج لحل هذه المشكلة بادارة عدد 12 لمبة فى الأركان الأربعـــة مـــن التقاطع منها أربعة باللون الأخضر Green, G ، واحدة فى كل ركن ، وأربعـــة باللون الأصفر Yellow, Y ، واحدة فى كل ركن ، وأربعة باللون الأحمر Red, R واحدة أيضا فى كل ركن كما هو مبين فى شكل (11-1) . لإنارة هذا العدد مـــن اللمبات (12) سنحتاج إلى بوابتى إخراج سنستخدم منهما 12 بت للمبات والأربـــع بتات الباقية سنتركها بدون استخدام حاليا أو لما قد يجد فى المستقبل من إضافـــات بالنسبة للدائرة . بالنسبة لمفتاح التحكم الذى سيشغل الإشارة إمـــا فـــى الوضـــع الطبيعى أو الوضع الترددى flashing فإننا سنحتاج لبوابة إدخال يقرأ منها المعالج قيمة هذا المفتاح باستمرار قبل البدء فى أى دورة من دورات الإشارة كما ســـنرى بعد قليل . لذلك فإننا سنحتاج للشرائح التالية لكى نتم عملية بناء دائرة التحكم فـــى إشارة المرور :

- شريحة المعالج Intel 8085 وقد وصلت جميع أطرافها إلى الجـــهد المناســـب (سواء أرضى أو Vcc) وسنرى كيفية توصيل هذه الأطراف فى الجزء القادم .

- شريحتين 74374 لفصل buffer مسار العناوين وقد رأينا ذلك فـــى الفصــل الثامن .
- شريحة 74245 لفصل buffer مسار البيانات كما فى الفصل الثامن أيضا .
- شريحة 74138 وشريحة 74125 للحصول على خطـــوط التحكـــم $\overline{MEMR}$ و $\overline{MEMW}$ و $\overline{IOR}$ و $\overline{IOW}$ وقد رأينا ذلك أيضا فى الفصل الثامن .
- ثلاث بوابات ، وهذه قد فضلنا أن نحصل عليها من الشـــريحة 8255A التـــى شرحناها فى الفصل العاشر .
- شريحة EPROM وهى الشريحة 2716 التـــى تحتـــوى علـــى 2 كيلوبـــايت EPROM حيث سيتم حرق (كتابة) البرنامج عليها .
- بعض الشرائح البسيطة مثل 7408 وهى AND وشريحة عاكس التى نحتاجـها لعملية التشفير المبسطة للبوابات وشريحة الذاكرة .
- القليل من المقاومات والمكثفات كما سنرى بعد قليل فى الدائرة الكاملة لـــهذا الكارت . سنطلق على هذه الدائرة إسم دائـــرة الميكروكومبيوتـــر ذى الكـــارت الواحد one board microcomputer .



شكل (11-1) إشارات المرور فى تقاطع رباعى

يجب على مستخدمى المعالج Z80 والمعالجات الأخـــرى أن يراعـــوا اســـتخدام الشرائح المناسبة كما رأينا فى الفصل الثامن عند فصل مسارات كل واحـــد مـــن هذه المعالجات .
عند تنفيذ هذا الكارت ستواجهنا بعض الأطراف لشريحة المعالج التى لا نعـــرف وظيفتها بالضبط لأننا لم ندرسها حتى الآن ولذلك فإننا لا نعلم ماذا نفعـــل بـــهذه الأطراف ، هل نتركها مفتوحة floating أم هل نوصلها بالأرضى أم Vcc. الجزء القادم من هذا الفصل سنشرح فيه وظيفة كل طرف من هذه الأطراف وماذا نفعل به على الكارت وذلك لكل واحد من المعالجين 8085 و Z80 وذلك قبل أن ندخـــل فى تفاصيل حل مثال إشارات المرور .

11-3 الأطراف الأخرى للمعالج 8085

لقد درسنا فى الأجزاء السابقة وظيفة أطراف المسارات الثلاثة للشـــريحة 8085 (40 طرفا) وهى كالتالى :

1. أطراف مسارى العناوين والبيانات وعددها 16 خط .
2. أطراف التحكم وعددها 3 خطوط ($\overline{WR}$, $\overline{RD}$,$IO/\overline{M}$).
3. الطرف ALE .
4. طرفان للقدرة Vcc والأرضى .

مجموع هذه الأطراف هو 22 طرفا ويتبقى 18 طرفا لم نعلم عنها شيئا حتى الآن وسنقوم فى هذا الجزء بشرح سريع لوظائف هذه الأطراف المتبقية . شكل (11-2) يبين إعادة لرسم أطراف الشريحة 8085 لتسهيل عملية المتابعة .

11-3-1 إشارات التزامن Clock signals

الشريحة 8085 تحتوى على مذبذب وهذا المذبذب يأخذ تردداته من بللـــورة أو كريستال crystal توصل بين الطرفين 1 و 2 للشريحة . هذا المذبذب ينتج عنـــه موجة جيبية ذات تردد يساوى 4 ميجاهرتز . الشريحة 8085 تحتاج إلى نبضـــات تزامن مربعة وذات تردد يساوى 2 ميجاهرتز ولذلك فإنه وكما هو موضح فـــى شكل (11-3) فقد وصل خرج المذبذب الجيبى على مقارن من نوع شميت ليقـــوم بتحويل الموجة الجيبية إلى موجة مربعة ثم بعد ذلك أدخلت هذه الموجة المربعـــة على قاسم ليقوم بقسمة تردد الموجة المربعة على 2 فنحصل عند خـــرج القاســـم على موجة مربعة ذات تردد 2 ميجاهرتز حيث تستخدم هذه الموجة فـــى جميـــع أغراض التزامن والتشغيل داخل الشريحة 8085 . شـــكل (11-3) يبيـــن كيفيـــة الحصول على هذه النبضات . هناك الكثير من الشـــرائح المحيطـــة والمســـاعدة للشريحة 8085 والتى تحتاج إلى نفس نبضات التزامن التى تعمل عليها ، لذلـــك

فقد تم إخراج نبضات التزامن على الطرف 37 لشريحة المعـــالج لكـــى تتمكـــن الشرائح والأجهزة المحيطة من الإستفادة منها .

Signal	Pin	Pin	Signal
X1	1	40	Vcc
X2	2	39	HOLD
Reset out	3	38	HLDA
SOD	4	37	CLK OUT
SID	5	36	$\overline{\text{Reset in}}$
TRAP	6	35	READY
RST7.5	7	34	IO/M
RST6.5	8	33	S1
RST5.5	9	32	$\overline{\text{RD}}$
INTR	10	31	$\overline{\text{WR}}$
$\overline{\text{INTA}}$	11	30	ALE
AD0	12	29	S0
AD1	13	28	A15
AD2	14	27	A14
AD3	15	26	A13
AD4	16	25	A12
AD5	17	24	A11
AD6	18	23	A10
AD7	19	22	A9
Vss	20	21	A8

المعالج 8085

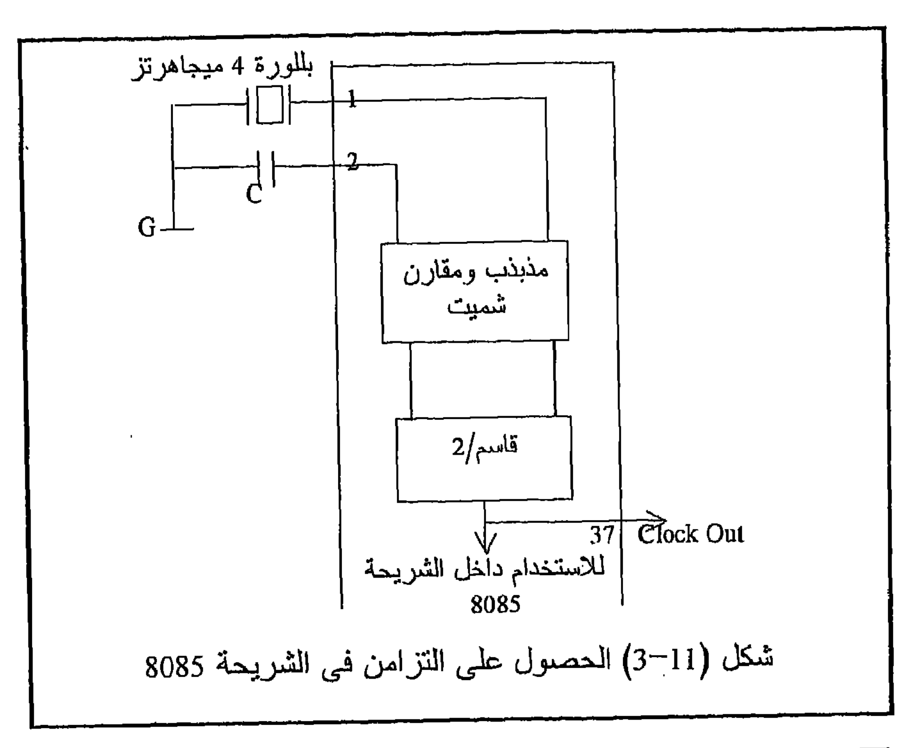
شكل (11-2) الشريحة 8085

11-3-2 بدأ وإعادة تشغيل المعالج

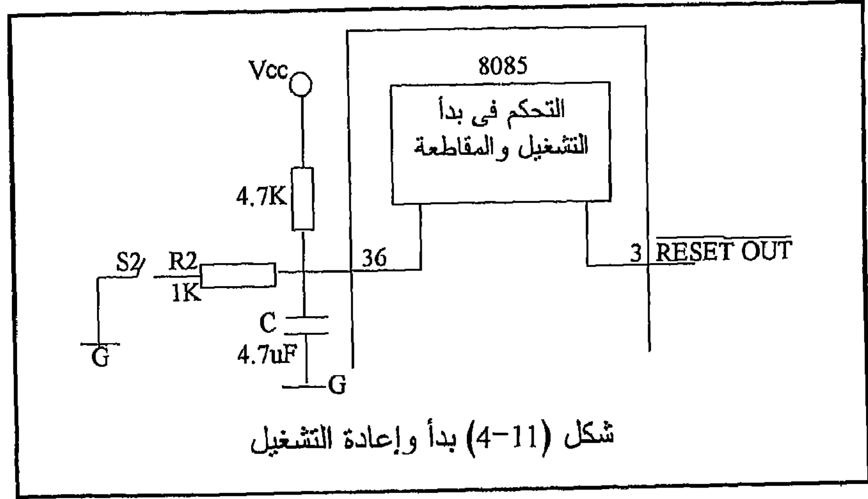
Starting and reseting the processor

عند وضع صفر على الطرف 36 للشريحة 8085 وهو الطرف $\overline{\text{RESET IN}}$ فـإن عداد البرنامج داخل الشريحة يصبح 0000H (ستعشرى) وعند عودة هذا الطـرف إلى الواحد فإن المعالج يبدأ تنفيذ البرنامج الخاص بإعادة التشغيل والموجود عنـد المكان 0000H فى الذاكرة . شكل (11-4) يبين دائرة يمكن استخدامها فى عمليـة بدأ وإعادة التشغيل . عند بدء التشغيل فإن المكثف C يشحن من خلال المقاومـة R1 إلى أن يصل جهده إلى القيمة high أو الواحد الثنائى عندها تبدأ الشريحة فـى العمل من العنوان 0000H كما ذكرنا . إن هذا التأخير الناتج من شحن المكثـف

يعتبر ضروريا حتى نضمن عدم تشغيل الشريحة قبل أن يستقر الجـــهد وحتـــى نتلافى المشاكل اللحظية Transients التى تحدث عند بدأ التشغيل .



شكل (11-3) الحصول على التزامن فى الشريحة 8085



شكل (11-4) بدأ وإعادة التشغيل

يمكن فى أى وقت أثناء عمل الشريحة 8085 إرسال إشارة إعادة تشغيل إليها مـــن خلال الطرف 36 عن طريق قفل المفتاح S2 حيث عندها يبدأ المكثـــف C فـــى

التفريغ من خلال المقاومة R2 وطالما أن المفتاح S2 مغلق فإن الشـــريحة 8085 لن تعمل . عند ترك المفتاح S2 يفتح يبدأ المكثف C في عملية الشحن من خلال المقاومة R1 حيث تبدأ خطوات إعادة التشغيل التي شرحناها سابقا . يمكن رؤيـــة عملية بدء وإعادة التشغيل على مبين الذبذبات كما في شكل (5-11) .

$\overline{RESET\ IN}$=0 $\overline{RESET\ IN}$ =1

الطرف ALE المعالج لا يعمل

شكل (5-11) الإشارة $\overline{RESET\ IN}$ وإشارة ALE على مبين ذبذبات

كما نعلم فإن الطرف ALE في أثناء عمل الشريحة 8085 يجب أن تكـــون عليـــه موجة مربعة تعكس حالة الإشارة الموجودة على أطراف المسارات AD0-AD7 ، فإذا كان هذا الخط low فإن الإشارة تكون بيانات ، أما إذا كان هذا الخـــط high فإن الإشارة تكون عناوين وكما نعلم فإن المعالج أثناء عمله لابد وأن يكون فـــي حالة تعامل مع بيانات أو عناوين لذلك فإننا يجب أن نرى موجة مربعـــة علـــى الخط ALE وهذه يمكن استخدامها للدلالة على أن المعالج يعمل . شـــكل (5-11) يبين الإشارة الموجودة على الطرف ALE مع الإشارة الموجودة علـــى الطـــرف $\overline{RESET\ IN}$ ، لاحظ في هذا الشكل أنه طالما أن طرف بدء التشـــغيل low فإنـــه ليس هناك أي إشارة على الطرف ALE وعند رجوع طرف بدء التشغيل إلـــى ال high تظهر الموجة المربعة على الطرف ALE دلالة على أن المعالج بـــدأ فـــي العمل .

11-3-3 الطرفان HOLD و HLDA

لكي يتم اتصال بين أي جهاز خارجي والذاكرة فإن ذلك يكون عادة عن طريـــق المعالج ، حيث يأخذ المعالج المعلومة من الجهاز الخارجي مـــن خـــلال مســـار البيانات ويقوم بإرسالها إلى المكان المحدد في الذاكرة من خلال مسار البيانـــــات وبمساعدة مساري العناوين والتحكم . هذه الطريقة من الإتصال تسمى بطريقـــة الإتصال غير المباشر مع الذاكرة حيث يكون الإتصال عن طريق المعالج . هناك طريقة الاتصال المباشر Direct Memory Access, DMA والتـــي يقـــوم فيـــها الجهاز الخارجي بإرسال معلوماته مباشرة إلى الذاكرة دون اللجوء إلى المعـــالج .

من أمثلة ذلك نسخ المعلومات من ذاكرة الحاسب إلى الإسطوانة اللينة Flopy disk والعكس . فى حالة الإتصال المباشر بالذاكرة فإن المعالج يجب عليه فى هذه الحالة أن ينعزل أو يترك أو ينفصل عن المسارات جميعها ليستخدمها الجهاز الخارجى فى عملية الإتصال بالذاكرة وحتى لا يحدث أى تصادم فى المعلومات على هذه المسارات . عندما يحتاج الجهاز الخارجى للمسارات ليقوم من خلالها بالإتصال المباشر بالذاكرة فإنه يخبر المعالج بذلك عن طريق اعطاء إشارة High على الخط HOLD . كلمة Hold تعنى امسك أو قف أو تجمد على وضعك الحالى وهذا هو ما يحدث فعلا للمعالج . عندما يتبين المعالج وجود high على الطرف HOLD وهو الطرف رقم 39 فإنه يقوم بإنهاء دورة الماكينة Machine cycle الحالية والتى يقوم بتنفيذها ثم يوقف تنفيذ البرنامج ويضع جميع خطوط المسارات فى حالة المقاومة العالية وهى حالة العزل أو الانفصال ويتجمد على هذا الوضع إلى أن يعود الخط HOLD إلى الصفر مرة أخرى . بذلك يكون المعالج قد انفصل عن المسارات ، وعند ذلك فإنه يعطى إشارة للدلالة على أنه انفصل عن المسارات عن طريق وضع الخط رقم 38 وهو HLDA فى وضع ال high أى واحد . HLDA تعنى الاعتراف بحالة التجمد Hold Acknowledge . المفروض على الجهاز الخارجى بعد أن يرسل الإشارة high إلى المعالج على خط ال HOLD أن يتتبع الإشارة الموجودة على الخط HLDA ، فعندما يصبح هذا الخط واحدا فإن ذلك يعنى أن المعالج قد انفصل عن المسارات وعندها فقط يستطيع الجهاز الخارجى أن يستخدم المسارات . يمكن استخدام الخط HLDA أيضا بحيث عندما يكون واحدا فإنه يضع جميع الشرائح المستخدمة فى عملية فصل المسارات الثلاثة فى الحالة الثالثة وهى حالة المقاومة العالية . فى دائرة الميكروكومبيوتر التى سنبنيها نريد أن يكون المعالج فى حالة عمل مستمر ولن نقاطعه أبدا أو نطلب منه مساراته ، ولذلك سنوصل خط الدخل HOLD بالحالة low أى غير فعال ، وخط الخرج HLDA سنتركه مفتوحا ولن نوصله بأى شىء لأنه يمثل إشارة خرج من المعالج وتركه مفتوحا لن يؤثر على عمل المعالج .

4-3-11 الطرف READY

إن جميع أجهزة الميكروكومبيتر تكون بها إمكانية تنفيذ البرامج بنظام الخطوة بخطوة Step by step execution فى حالة تعاملها بلغة الأسمبلى . فى هذا النظام فإن المعالج ينفذ خطوة أو أمرا واحدا من البرنامج بعد إعطائه الأمر بذلك ثم ينتظر أمرا آخر لكى ينفذ الخطوة التالية وهكذا . إن المعالج بعد تنفيذ أى أمر يدخل فى حالة انتظار Waiting إلى أن يجيئه الأمر بتنفيذ الخطوة التالية . فى أثناء حالة الانتظار تبقى آخر إشارة وضعت على المسارات كما هى ولا تتغير حتى أنه بعد الإنتهاء من حالة الانتظار يستأنف المعالج عملية التنفيذ من نفس

المكان الذى توقف عنده . السؤال الآن كيف نستطيع إدخال المعالج فى حالة الانتظار هذه؟ إن ذلك يتم عن طريق الطرف رقم 35 من الشريحة 8085 وهو الطرف READY بمعنى جاهز أو مستعد . إن المعالج قبل تنفيذ أى أمر يقوم باختبار الطرف READY فإن كان هذا الطرف high يتم تنفيذ الأمر وإن كان الطرف READY فى حالة low فإن المعالج لن يتم تنفيذ الأمر وسيدخل فى حالة انتظار كما شرحنا . يجب هنا أن نفرق بين حالة الانتظار الناتجة من صفر يوضع على الطرف READY وحالة ال HOLD التى رأيناها فى الجزء السابق . فى حالة HOLD ينفصل المعالج تماما عن المسارات وتكون جميع المسارات فى وضع المقاومة العالية . أما فى حالة الانتظار التى نحن بصددها هنا فلا ينفصل المعالج عن المسارات ولكن تبقى المسارات حاملة لآخر إشارة تم وضعها على المسارات قبل أن يكون الخط READY صفرا . فى دائرة الميكروكومبيوتر التى سنبنيها نريد المعالج أن يكون فى حالة عمل مستمر ولذلك فسوف نضع الخط READY فى حالة HIGH باستمرار أى غير فعال .

5-3-11 طرفى الحالة S1, S0

الإشارة الموجودة على هذين الخطين تبين حالة المعالج عند أى لحظة من اللحظات حيث أن المعالج لابد وأن يكون فى حالة من الأربع حالات الآتية :

1. حالة انتظار Waiting وهذه كما رأينا تكون عندما نضع صفرا أو low على الخط READY .
2. حالة كتابة سواء كانت كتابة فى ذاكرة أو بوابة إخراج .
3. حالة قراءة أمر من الذاكرة وتكون هذه فى أثناء دورة إحضار الأمر .
4. حالة قراءة معلومة سواء كانت المعلومة فى الذاكرة أو فى بوابة إدخال .

شكل (11-6) يبين الشفرة الموجودة على هذين الخطين فى مقابل كل حالة من الحالات الأربع . لاحظ أن الإشارة الموجودة على هذين الخطين تكون خارجة من المعالج ولذلك فإننا فى دائرة الميكروكومبيوتر التى سنبنيها سنترك هذين الخطين مفتوحين ولن نوصلهما بأى توصيلات خارجية .

S0	S1	حالة المعالج
0	0	انتظار Waiting
1	0	كتابة Write
0	1	قراءة Read
1	1	قرائة أمر Fetching

شكل (11-6) الحالات المختلفة للشريحة 8085

11-3-6 أطراف المقاطعة INTR, TRAP, RST7.5, RST6.5, RST5.5

يمكن استخدام خطوط المقاطعة لإيقاف المعالج من تنفيذ البرنامج الذى يقوم بتنفيذه الآن وجعله يذهب لتنفيذ برنامج آخر يسمى برنامج خدمة المقاطعة وبعد الانتهاء من تنفيذ برنامج خدمة المقاطعة يعود المعالج إلى البرنامج الأصلى حيث يستأنف تنفيذه من نفس المكان الذى حدثت عنده المقاطعة . لقد أفردنا فصلا كاملا للمقاطعة لمن يريد تفاصيل وأمثلة عن هذا الموضوع ، وما يهمنا الآن هو ماذا سنفعل بهذه الخطوط الآن ؟ إذا كنا نريد استخدام المقاطعة فى البرنامج الذى سنستخدمه مع الميكروكومبيوتر الذى ننوى بناءه فى هذا الفصل فعلينا أن نرجىء عملية بناء الدائرة لحين مراجعة الفصل الخاص بالمقاطعة ، أما إذا كنا لن نستخدم المقاطعة (كما هى الحال فى المثال الذى نحن بصدده) فإن جميع هذه الخطوط يجب أن توضع فى حالة عدم الفعالية وهذه الحالة تكون عند توصيل هذه الخطوط بالأرضى أى low كما هو مبين فى الدائرة الكاملة للميكروكومبيوتر فى آخر هذا الفصل .

11-3-7 الطرفان SOD, SID

الشريحة 8085 يمكنك أن تستقبل منها أو ترسل إليها بيانات تتابعية Serial أى بت بعد بت وليس بنظام البايت كما عرفنا سابقا . لن نتكلم عن عملية إدخال وإخراج البيانات تتابعيا فى هذا الكتاب ولكن الذى يهمنا هنا هو أن نعرف أن البيانات التتابعية تخرج من المعالج على الطرف SOD الذى يعنى Serial Output Data بعد وضعها فى مسجل التراكم وأما البيانات المراد إدخالها تتابعيا فإنها تدخل على الطرف SID الذى يعنى Serial Input Data ومنه إلى مسجل التراكم.
هذان الطرفان لن يكون لهما أى فعالية فى دائرة الميكروكومبيوتر ذى الكارت الواحد ولذلك فإننا سنتركهما مفتوحان حيث أنهما لن يؤثرا على تشغيل المعالج بأى حال . لاحظ أن جميع الخطوط التى تحمل إشارة خرج من المعالج والتى لن يتم استخدامها يجب أن نتركها مفتوحة ولا يتم توصيلها بأى شىء (الأرضى أو ال Vcc) لأن ذلك يمكن أن يؤثر على عمل المعالج بما لا تحمد عقباه .

11-4 الأطراف الأخرى للمعالج Z80

11-4-1 أطراف المقاطعة $\overline{RESET}, \overline{BUSRQ}, \overline{INT}, \overline{NMI}$

جميع هذه الأطراف تعتبر مداخل للالمعالج يمكن مقاطعته من عليها وجميعها فعالة عند الصفر active low ، ولذلك فإنه طالما أن دائرة الميكروكومبيوتر ذى الكارت الواحد لن تستخدم المقاطعة أصلا فإننا سنضع جميع هذه الخطوط فى

حالة خمول بتوصيلها إلى Vcc أى high مع العلم أن فصل المقاطعة فـــى هــذا الكتاب خاص بعمليات المقاطعة لمن يريد الإستزادة فى هذا الموضـــوع . الــذى يهمنا معرفته هنا هو العنوان الذى يقفز إليه المعالج عند عمل RESET له أو عند إعادة القدرة إليه ؟ عند إعطاء نبضة على الطرف $\overline{RESET}$ يقفز المعـــالج Z80 إلى العنوان 0000H حيث يبدأ تنفيذ البرنامج الموجود فى هذا المكان ، لذلك يمكن توصيل هذا الطرف بدائرة كالمبينة فى شكل (11-4) والتى اســتخدمناها لعمــل RESET للبروسيسور 8085 . الخطان $\overline{BUSRQ}$ و $\overline{BUSACK}$ يكافئان الخطـــان HOLD و HLDA فى حالة المعالج 8085 حيث يمكن لأى جهاز خارجى بجعــل الطرف $\overline{BUSRQ}$ فعالا low أن يطلب من المعالج التخلى عن المسارات ووضعها فى الحالة المنطقية الثالثة حتى يتسنى للجهاز الخارجى إستعمالها فـــى التعـــامل المباشر مع الذاكرة مثلا . BUSRQ تعنى طلب المسارات Bus Requist. عندمـــا يتخلى المعالج عن المسارات يقوم بوضع صفـــر low علـــى الخـــط $\overline{BUSACK}$ والذى منه يعرف الجهاز الخرجى أن المعالج قد تخلى فعــــلا عــن المســـارات وتركها وكلمة BUSACK تعنى اعترافا بطلب التخلـــى عـــن المســـارات Bus Acknowledge . فى حالة استخدام المعالج Z80 فـــى دائـــرة الميكروكومبيوتـــر المقترحة يجب أن يوضع الخط $\overline{BUSRQ}$ فى حالة خمول أو عدم فعالية بتوصيلــة على Vcc أى high . أما الخط $\overline{BUSACK}$ فطالما أنه خط خرج فيجب أن يـــترك مفتوحا open ولا يوصل بشىء .

11-4-2 الطرف RFSH

يستخدم هذا الخط فى عملية تجديد محتويـــات الذاكـــرة الديناميكيـــة dynamic memory كل فترة زمنية محددة ، وطالما أن هذا الخط يعتبر خط خرج أى يحمل إشارات خارجة من المعالج فإننا سنتركه مفتوحا طالما أننا لن نستخدم هذا النـــوع من الذاكرة فى الدائرة المقترحة ولا ننوى الدخول فى تفاصيل عملية المواجهة مع الذاكرة الديناميكية هنا .

11-4-3 الطرف HALT

الخط HALT خط خرج ويكون فعالا عندما يكون المعالج فى حالـــة HALT أى توقف عن العمل بعد تنفيذ الأمر HALT ولا يخرج المعالج من هذه الحالـــة إلا عن طريق مقاطعة وطالما أنه خط خرج فسنتركه مفتوحا open .

11-4-4 الطرف $\overline{WAIT}$

يستخدم هذا الخط فى إدخال المعالج فى حلقة إنتظار حيث يتوقف فيها المعـــالج عن تنفيذ أى أمر ولا ينفصل عن المسارات ويمكن استخدام هذا الطرف فى تنفيذ

البرامج بنظام الخطوة خطوة مثل الطرف READY فى المعـــالج 8085 وأيضـــا عندما تكون الأجهزة المحيطة غير مستعدة للتعامل مع المعـــالج بســبب فــارق السرعة مثلا . هذا الخط يكون فعالا عندما يكون صفـــرا أى أنــه active low ولذلك فإننا يجب أن نوصله على Vcc فى دائرة الميكروكومبيوتـــر المقترحــة لنجعله غير فعال .

11-4-5 الطرف M1

هذا الخط هو خط خرج يبين حالة المعالج حيث يكون فعالا عندما يكون المعـــالج فى حالة إحضار الأوامر من الذاكرة وهذا الخط لن نستخدمه فى دائرتنا ولذلـــك سنتركه مفتوحا open .

11-4-6 الطرف CLK

يستخدم هذا الخط لإدخال نبضات التزامن الخاصة بالمعالج وهى نبضــات TTL يتراوح ترددها بين 2.5MH أو 4MH أو 6MH وذلك على حسب نوع المعــــالج المستخدم .

بذلك نكون قد انتهينا من التعرف على جميع الأطراف الأخـــرى للبروسيســـور Z80 وعرفنا فكرة موجزة عنها ونستطيع أن نلخص القول فى أن جميع خطـــوط الخرج الغير مستخدمة يجب أن تترك مفتوحة open وجميع خطوط الدخل الغـــير مستخدمة توضع فى وضع خمول أى عدم فعالية .

11-5 إشارات المرور

الآن بعد أن عرفنا وظيفة كل طرف من أطراف المعالج وماذا سنفعل بكل طرف من الأطراف الغير مستخدمه ، ماذا عن دائرة المعالج التى ستتحكم فـــى إشـــارة المرور كما ذكرنا فى المثال التوضيحى ؟ قبل أن ندخل فى تفاصيل بناء أو حـــل هذه المسألة سننفق بعض الوقت فى التفكير فى خطة الحل . إن حل هذه المســـألة كباقى المسائل التى تستخدم المعالج يتكون من جزء بناء الدائرة hardware وجزء برمجة software . بالنسبة لجزء البناء وكما أشرنا فى بداية الفصل فإننا ســنحتاج لما يلى :

1. المعالج وقد تم فصل جميع مساراته ، مسار البيانات D0 إلـــى D7 ومســـار العناوين A0 إلى A15 ومسار التحكم بالإضافة إلى خط إعادة الوضع . سنستخدم كما ذكرنا المعالج 8085 كمثال ومن يريد استخدام أى معالج آخر فذلك أصبــــح سهلا جدا بعد قراءة هذا الفصل . ولقد رأينا فى الفصل الثـــامن كيفيـــة فصـــل

المسارات الثلاثة ويمكن مراجعة ذلك وبالذات للمعالج 8085 الذى سنستعمله فى هذا المثال . شكل (11-7) يبين المعالج والشرائح المستخدمه فى عملية الفصل .

2. إشارة المرور الرباعية كما رأينا بها 12 لمبة (ثلاثة فى كل ركن من أركان التقاطع ، أحمر وأخضر وأصفر) ومطلوب إدارة هذه اللمبات بتتابع معين وأزمنة محسوبة كما سنرى بعد قليل . لذلك سنحتاج إلى بوابتى إخراج بمجموع 16 بت سنستخدم منهم 12 لإدارة ال 12 لمبة ويتبقى 4 بتات من ال 16 لن تستخدم وستترك للاستخدام المستقبلى . كما ذكرنا سابقا فى المثال التوضيحى فإن هناك مفتاح سيقوم المعالج بقراءته دائما وإذا كان هذا المفتاح واحدا (ON) فإن الإشارة تعمل فى الحالة العادية ، وأما إذا كان المفتاح صفرا (OFF) فإن الإشارة ستعمل فى الحالة الترددية للون الأصفر flashing . لذلك فإننا سنحتاج إلى بوابة إدخال سنستعمل منها بت واحدة لقراءة حالة هذا المفتاح والباقى لن يستخدم وسيترك لأى استخدامات مستقبلية . خلاصة القول أننا سنحتاج إلى بوابتى إخراج وبوابة إدخال . بعد أن قررنا أننا سنحتاج إلى بوابتى إخراج وبوابة إدخال يجب أن نقرر أيضا بأى طريقة من الطرق التى درسناها فى الفصل العاشر سنبنى هذه البوابات . لقد قررنا نحن أن نستخدم البوابات القابلة للبرمجة أى الشريحة 8255A وذلك لسهولة برمجتها وبنائها وكونها شريحة واحدة تحتوى الثلاث بوابات التى نحتاج إليها . شكل (11-8) يبين طريقة توصيل هذه الشريحة على المسارات الثلاثة القادمة من المعالج .

3. يبين شكل (11-8) أيضا كيفية توصيل شريحة الذاكرة EPROM وهى الشريحة 2716 التى سنكتب عليها برنامج التحكم فى كل العملية . لقد درسنا فى الفصل التاسع طريقة توصيل الذاكرة على المعالج والعملية هنا أبسط بكثير مما شرحنا فى الفصل التاسع ، فإنه طالما أن المعالج سيكون متصلا بشريحة ذاكرة واحدة فإن عملية العنونة والتشفير من الممكن أن تكون بسيطة جدا إذا وصلنا خط تنشيط الشريحة 2716 وهو الخط $\overline{CS}$ على خط التحكم $\overline{MEMR}$ وخط تنشيط خرج الشريحة $\overline{OE}$ بالأرضى مباشرة . فى هذه الحالة فإن الشريحة ستعمل وتخرج خرجها مع أى أمر قراءة من الذاكرة ولا حرج فى ذلك حيث أنها هى شريحة الذاكرة الوحيدة الموصلة مع المعالج . لاحظ أنه طالما أن ال EPROM ستعمل مع أى عنوان يخرجه المعالج فإنه يجب أن نتوقع أنها ستعمل بمجرد توصيل القدرة للدائرة لأن المعالج 8085 كما رأينا سابقا عند توصيل القدرة إليه أو إعادة وضعه RESET فإنه يبدأ التنفيذ من العنوان 0000H حيث سيشغل ال EPROM .

4. يمكن توصيل كل بت من بتات الخرج الخارجة من الشريحة 8255 على قاعدة ترانزستور ليعمل كفاصل buffer لإدارة هذه الدايودات حتى تعطى كمية إضاءة أحسن . شكل (11-8) يبين ذلك أيضا .

5. يمكن توصيل خرج البوابات على لمبات 220 فولت للحصول على إنارة قوية ودائرة واقعية عن طريق استخدام عوازل ضوئية opto—couplers .

بالنسبة لجزء البرمجة software فإننا سنفكر فيه بالطريقة التالية :

1. يحتوى شكل (11–9) على جدول به جميع الحالات الممكنة لجميع اللمبات وزمن كل حالة والشفرة أو الرقم المطلوب إخراجه على بوابات الإخراج للحصول على هذه الحالة . من شكل (11–9) نلاحظ مثلا أنه فى الحالة الأولى يكون الأخضر الأول مضيئا والأحمر فى جميع الاتجاهات الأخرى مضيئا أيضا والأصفر فى جميع الاتجاهات مطفأ وسوف يستمر هذا الوضع لمدة 50 ثانية . لاحظ أن المضيء فى الجدول يساوى واحدا والمطفأ يساوى صفرا ، لذلك فإن هذا الوضع الأول سيكافيء الشفرة 4C على بوابة الإخراج الأولى والشفرة 02 على بوابة الإخراج الثانية . بعد ذلك يضيء الأصفر الأول مع الأخضر الأول مع نفس الوضع السابق لجميع اللمبات الأخرى وذلك لمدة 10 ثوان وذلك كزمن تحذير بالوقوف بعد الضوء الأخضر . بعد ذلك ستضيء جميع اللمبات الحمراء فى جميع الاتجاهات لمدة 5 ثوان كزمن أمان خوفا من السيارات المسرعة التى لا تستطيع التوقف فى خلال الضوء الأصفر . بعد ذلك تكرر هذه الأزمنة لجميع الاتجاهات الأخرى .

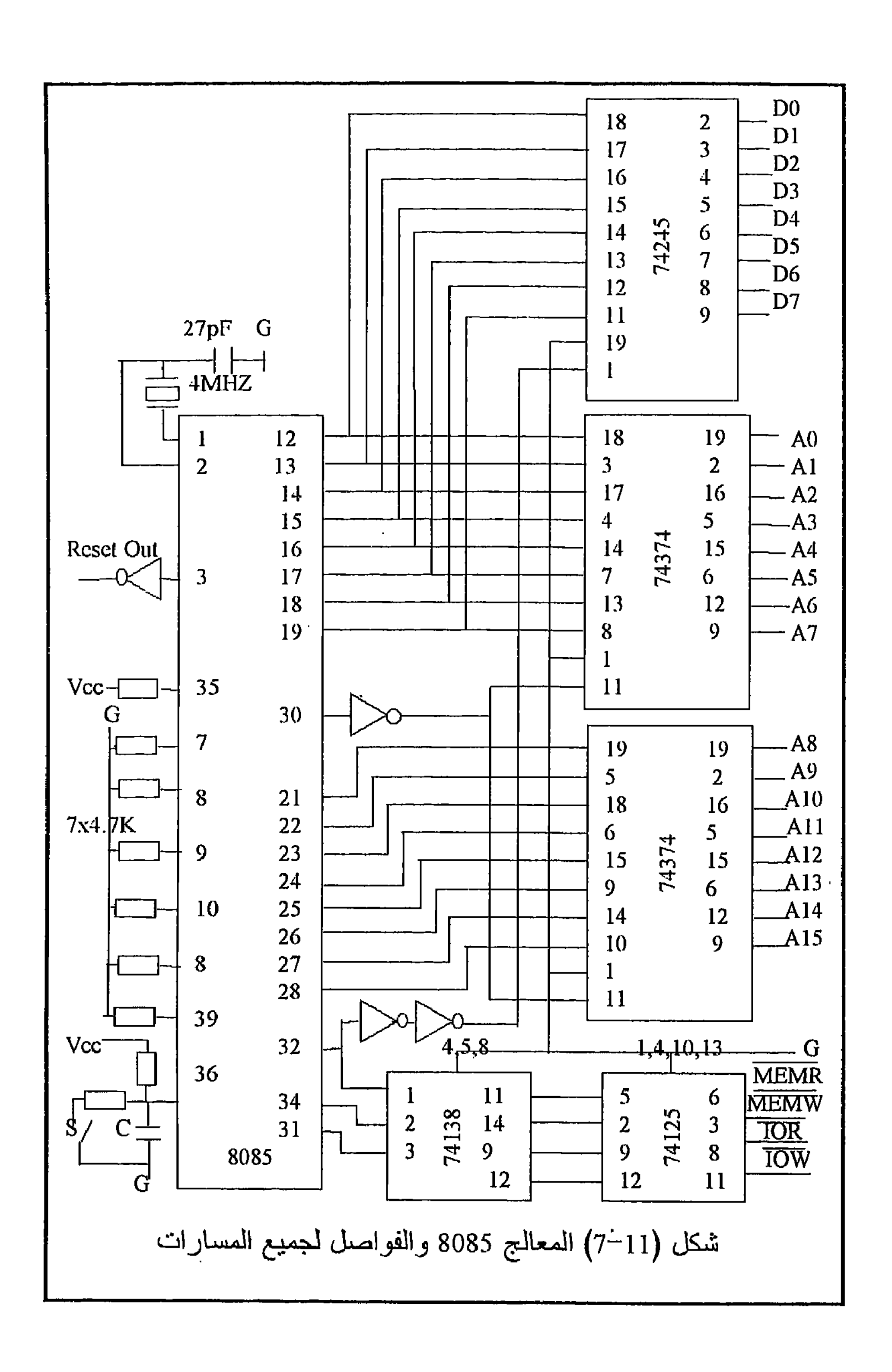
لاحظ أن هذه الأزمنة يمكن التحكم فيها بالزيادة والنقصان على حسب الرغبة ، ثم إن تتابع الضوء الأخضر فى الاتجاهات المختلفة بمعنى أن أى اتجاه سيسمح له بالمرور وأى اتجاه سيسمح له بعده ، هذه أيضا يمكن التحكم فيها . إن فكرة البرنامج التى نقترحها هنا (بالطبع فإن كل قارىء ستكون لديه فكرة مختلفة وربما أفضل) تعتمد على ما يلى :

1. سنخزن فى مكان ما فى الذاكرة الشفرات المطلوب إخراجها على بوابات الإخراج لكل حالة وكذلك زمن كل حالة بالتتابع من الجدول المبين فى شكل (11–9) وليكن ذلك كما يلى :

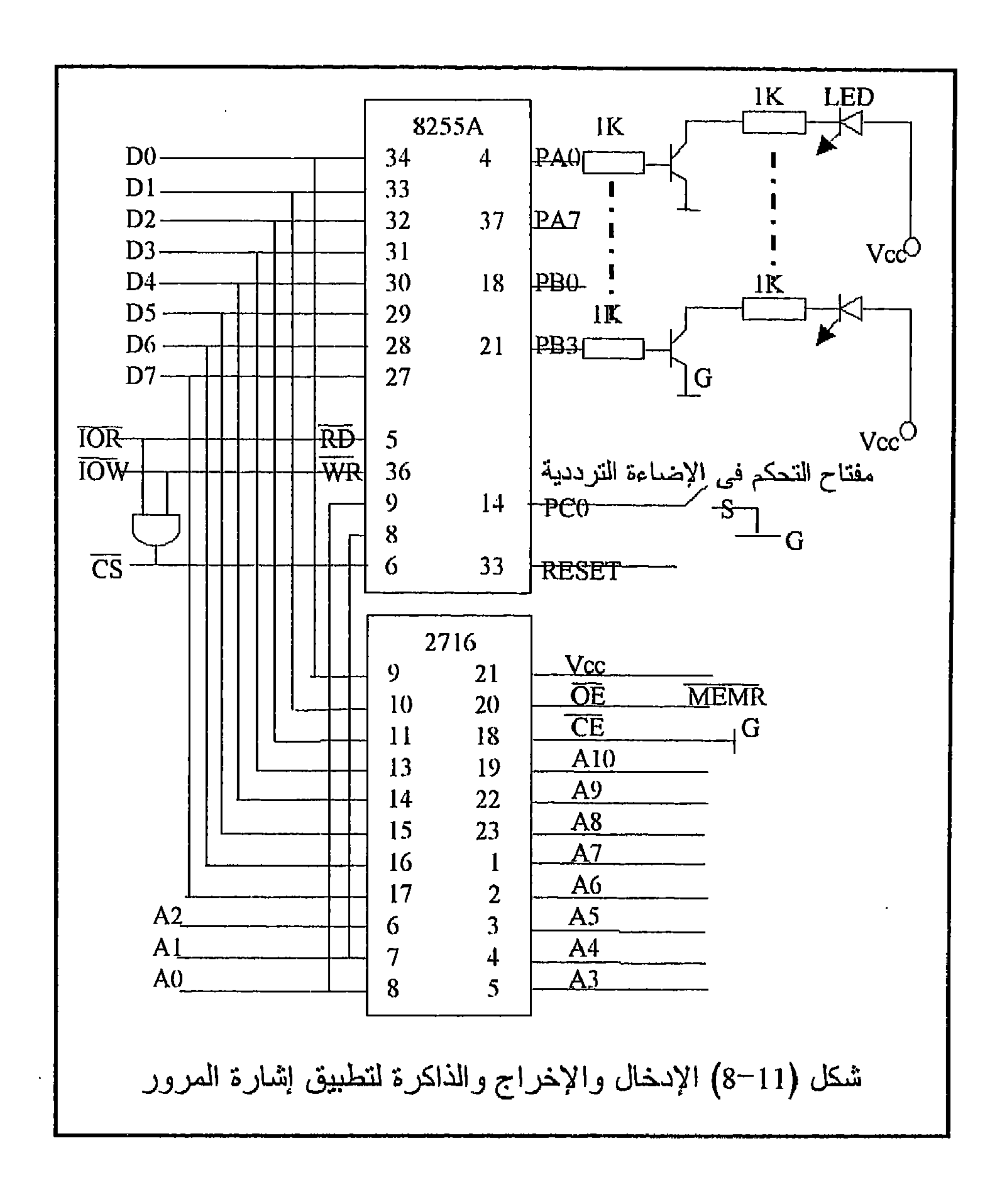
E100 4C , 4C ستخرج على البوابة الأولى
E101 02 , 02 ستخرج على البوابة الثانية
E102 50 , 50 زمن تأخير للحالة الأولى
E103 4E , 4E تخرج على البوابة الأولى
E104 02 , 02 تخرج على البوابة الثانية
E105 10 , 10 هى زمن تأخير للحالة الثانية

وهكذا سيحتوى هذا الجدول على 12 بايت .

2. سيكون البرنامج عبارة عن قراءة للشفرتين المقابلتين لكل حالة وإخراجهما على بوابتى الإخراج رقم 00 ورقم 01 ثم الدخول فى زمن التأخير المقابل لهذه الحالة .



شكل (11-7) المعالج 8085 والفواصل لجميع المسارات



شكل (11-8) الإدخال والإخراج والذاكرة لتطبيق إشارة المرور

3. قبل الدخول فى أى حالة جديدة لابد أن يقرأ المعالج بوابة الإدخال رقم 02 ليعرف إذا كان مفتاح التحكم المقابل للبت رقم صفر يساوى واحد أم صفر ، فإذا كانت هذه البت تساوى واحد فإن البرنامج يسير فى سيره الطبيعى كما فى الجدول المبين فى شكل (11-9) ، وإذا كانت هذه البت تساوى صفرا سيقفز إلى برنامج آخر ينفذ عملية التردد على اللون الأصفر .

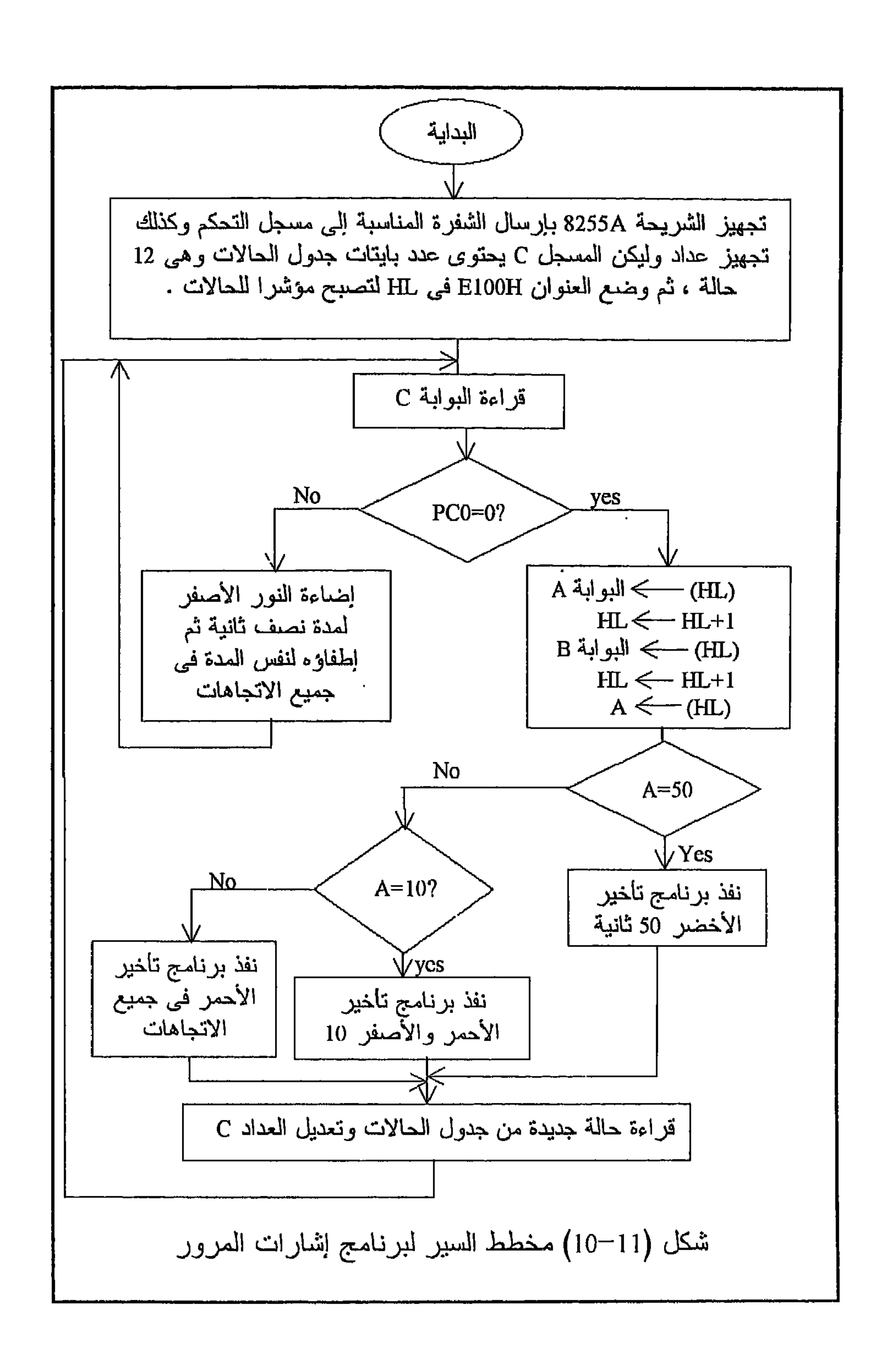
شكل (11-10) يبين خريطة التدفق ، والبرنامج مكتوبا بلغة الأسمبلى للبروسيسور 8085 سيكون كما يلى : (يلاحظ من هذا البرنامج خلوه من البرامج

الفرعية subroutines بالرغم من أفضلية استخدامها هنا فى مثل هذه التطبيقات ولكننا تجنبنا ذلك لسببين أولهما أننا مازلنا لم نشرح البرامج الفرعية حتى الآن فى هذا الكتاب وثانيهما أن الدائرة التى بنيناها ليس بها ذاكرة RAM تستخدم كمكدسة stack فى حالة استخدام البرامج الفرعية كما سنرى) .

الزمن بالثانية	البوابة 01 00	G4Y4R4	G3Y3R3	G2Y2R2	G1Y1R1	رقم الحالة
50	02 4C	0 0 1	0 0 1	0 0 1	1 0 0	1
10	02 4E	0 0 1	0 0 1	0 0 1	1 1 0	2
5	02 49	0 0 1	0 0 1	0 0 1	0 0 1	3
50	02 61	0 0 1	0 0 1	1 0 0	0 0 1	4
10	02 71	0 0 1	0 0 1	1 1 0	0 0 1	5
5	02 49	0 0 1	0 0 1	0 0 1	0 0 1	6
50	03 09	0 0 1	1 0 0	0 0 1	0 0 1	7
10	03 89	0 0 1	1 1 0	0 0 1	0 0 1	8
5	02 49	0 0 1	0 0 1	0 0 1	0 0 1	9
50	08 49	1 0 0	0 0 1	0 0 1	0 0 1	10
10	0C 49	1 1 0	0 0 1	0 0 1	0 0 1	11
5	02 49	0 0 1	0 0 1	0 0 1	0 0 1	12
5	02 49	من هنا يبدأ تكرار هذه الحالات				1

شكل (11-9) جدول الحالات المختلفة لإشارة المرور

```
البوابة A أخراج و B أخراج و C إدخال والجميع فى الحالة صفر
        MVI A,89H   ;
        OUT 03H     ;   إرسال إلى مسجل التحكم فى 8255A
START:  MVI C,0CH   ;   مسجل C عداد يحتوى 12 بايت
        LXI H,E100  ;   HL مؤشر للذاكرة ابتداء من E100
READ:   IN 02H      ;   قراءة البوابة C
        ANI 01H     ;   حجب لجميع البتات ما عدا الأولى
        JZ FLASH    ;   قفز عندما PC0=0 لترذذ اللون الأصفر
        MOV A,M     ;   إحضار الشفرة الأولى من الذاكرة
        OUT 00      ;   إخراج هذه الشفرة على البوابة A
        INX H       ;
        MOV A,M     ;   إحضار الشفرة الثانية من الذاكرة
        OUT 01      ;   إخراج هذه الشفرة على البوابة B
        INX H       ;
```



شكل (11–10) مخطط السير لبرنامج إشارات المرور

```
         MOV A,M       ;   إحضار زمن تأخير
         CPI 50H       ;   هل هذا الزمن =50
         JNZ TEN       ;   قفز إذا لم يكن الزمن=50
         MVI B,6CH     ;   بداية برنامج تأخير 50 ثانية
LOOP1:   MVI D,FFH     ;
LOOP2:   MVI A,FF      ;
LOOP3:   DCR A         ;
         JNZ LOOP3     ;
         DCR D         ;
         JNZ LOOP2     ;
         DCR B         ;
         JNZ LOOP1     ;
         JMP CONTIN    ;
TEN:     CPI 10H       ;   هنا التأخير لا يساوى 50 فهل =10
         JNZ FIVE      ;   إقفز إذا لم يكن التأخير = 10
         MVI B,16H     ;   بداية برنامج تأخير 10 ثوان
LOOP4:   MVI D,FFH     ;
LOOP5:   MVI A,FFH     ;
LOOP6:   DCR A         ;
         JNZ LOOP6     ;
         DCR D         ;
         JNZ LOOP5     ;
         DCR B         ;
         JNZ LOOP4     ;
         JMP CONTIN    ;
FIVE:    MVI B,0BH     ;   هنا التأخير لا يساوى 10 ولكن يساوى 5 ثوان
LOOP7:   MVI D,FFH     ;   بداية برنامج تأخير 5 ثوان
LOOP8:   MVI A,FFH
LOOP9:   DCR A
         JNZ LOOP9     ;
         DCR D         ;
         JNZ LOOP8     ;
         DCR B         ;
         JNZ LOOP7     ;
CONTIN:  DCR C
         JZ START      ;
         INX H         ;
         JMP READ      ;
FLASH:   MVI A,92H     ;   بداية برنامج تردد الضوء الأصفر
```

```
         OUT 00H      ;      الرقمان 92H و 04H يضيئان الأصفر
         MVI A,04H    ;      فى جميع الأتجاهات
         OUT 02H      ;
         MVI B,D8H    ;      بداية تأخير نصف ثانية
LOOPA:   MVI D,FFH    ;
LOOPB:   DCR D        ;
         NOP          ;
         JNZ LOOPB    ;
         DCR B        ;
         JNZ LOOPA    ;
         MVI A,00H    ;      إطفاء النور الأصفر فى جميع الأتجاهات
         OUT 00H      ;
         OUT 01H      ;
         MVI B,D8H    ;      بداية تأخير نصف ثانية
LOOPC:   MVI D,FFH    ;
LOOPD:   DCR D        ;
         NOP          ;
         JNZ LOOPD    ;
         DCR B        ;
         JNZ LOOPC    ;
         JMP READ ;          قفز لقراءة بوابة الإدخال PC
```

- سنرى فى فصل البرامج الفرعية كيفية الحصول على أزمنة التأخير باستخدام البرامج الفرعية مما سيؤدى إلى اختصار خطوات مثل هذا البرنامج بدرجة كبيرة ، ويمكن إعادة كتابة البرنامج بهذا الوضع كتمرين على البرامج الفرعية .

11-6 تـمــاريــن

1. أعد تصميم كارت الميكروكومبيوتر ، وكذلك البرنامج ، الخاصين بالتحكم فى إشارة المرور ولكن مستخدما هذه المرة المعالج Z80 بدلا من المعالج 8085 كما كان فى هذا الفصل .

الفصل الثانى عشر

البرامج الفرعية

Subroutines

12-1 مقدمة

نستطيع القول بأنه عامة عندما تواجهك كمبرمج مشكلة كبيرة ومطلوب منك برمجتها فإن أسهل الطرق لذلك هى أن تقوم بتجزىء أو تكسير هذه المشكلة الكبيرة إلى مشاكل أو مسائل أصغر ثم تقوم ببرمجة هذه المسائل الصغيرة كل على حدة ثم يكون هناك برنامج أساسى يقوم بتجميع أو تنفيذ هذه الأجزاء الصغيرة بالتتابع الذى يحل المسألة أو المشكلة الأساسية . أحد طرق التجزىء هذه هى البرامج الفرعية subroutines . إن استخدام البرامج الفرعية من مميزاتها تسهيل عملية البرمجة واختصار كمية الذاكرة المستخدمة لكتابة شفرات البرنامج كما سنرى فى هذا الفصل .

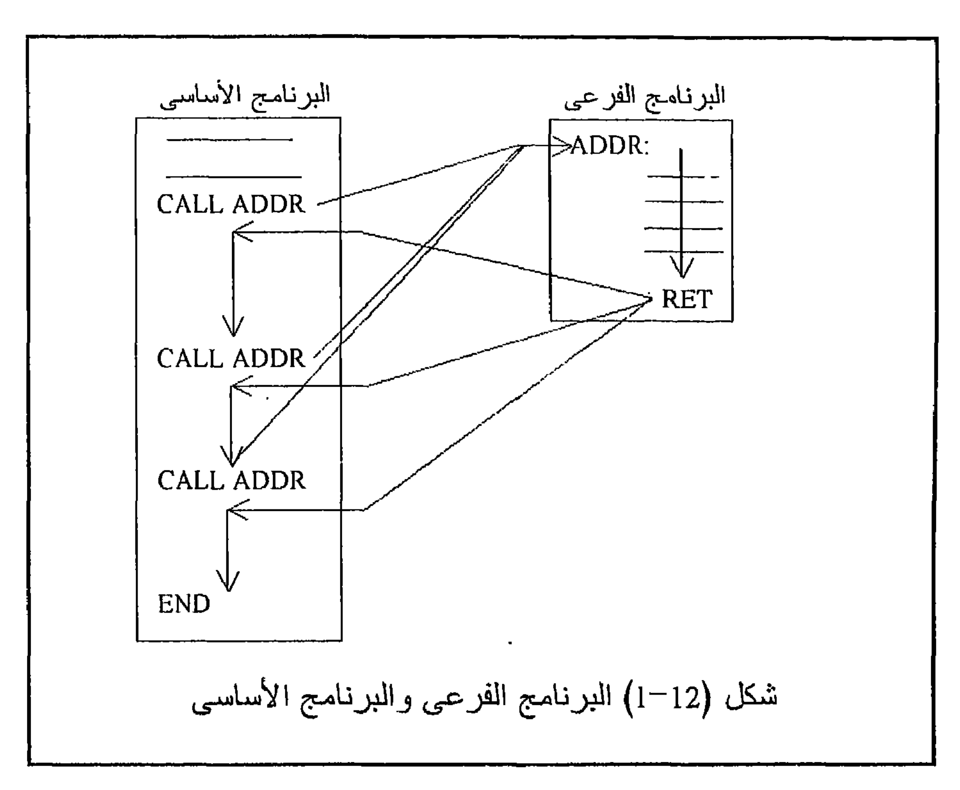
12-2 ما هو البرنامج الفرعى ؟

شكل (12-1) يبين رسما توضيحيا لعلاقة البرنامج الفرعى بالبرنامج الأساسى. نلاحظ من هذا الشكل أن البرنامج الفرعى عبارة عن جزء من برنامج ، أو برنامج صغير ، يتم النداء عليه للتنفيذ من البرنامج الأساسى فينفذ ، وبعد الانتهاء من تنفيذه تتم العودة إلى البرنامج الأساسى وعند نفس المكان الذى تم الخروج منه للبرنامج الفرعى . أى أن طريقة تنفيذ البرنامج الفرعى تشابه وإلى حد كبير طريقة تنفيذ أوامر القفز ، الاختلاف فقط هو فى عملية العودة إلى نفس المكان الذى تم القفز منه فى البرنامج الأساسى بعد الانتهاء من تنفيذ البرنامج الفرعى ، ويرجع ذلك إلى بعض الخطوات أو الاحتياطات التى يعملها المعالج قبل القفز إلى البرنامج الفرعى .

من شكل (12-1) نستطيع أن نتبين الفائدة العظيمة من استخدام البرامج الفرعية وهى توفير الذاكرة المستخدمة لكتابة البرنامج . فى الكثير من التطبيقات يكون هناك جزء من البرنامج تكون مضطرا لكتابته أكثر من مرة وكمثال على ذلك جزء البرنامج الذى يعمل زمن التأخير فى برنامج إشارات المرور فى الفصل السابق . إن مثل هذا الجزء باستخدام البرامج الفرعية يمكن كتابته مرة واحدة فقط وفى كل مرة تكون هناك الحاجة إلى تنفيذه يتم النداء عليه بالأمر CALL فينفذ وبعد الانتهاء من تنفيذه ترجع عملية التنفيذ إلى حيث خرجت من البرنامج الأساسى .

شكل (12-2) يبين خاصية أخرى فى البرامج الفرعية وهى أن أى برنامج فرعى يمكنه النداء على برنامج فرعى آخر ، فمثلا البرنامج الأساسى ينادى البرنامج الفرعى (أ) والبرنامج الفرعى (أ) ينادى البرنامج الفرعى (ب) والبرنامج الفرعى (ب) ينادى البرنامج الفرعى (ج) وهكذا لأى عدد من

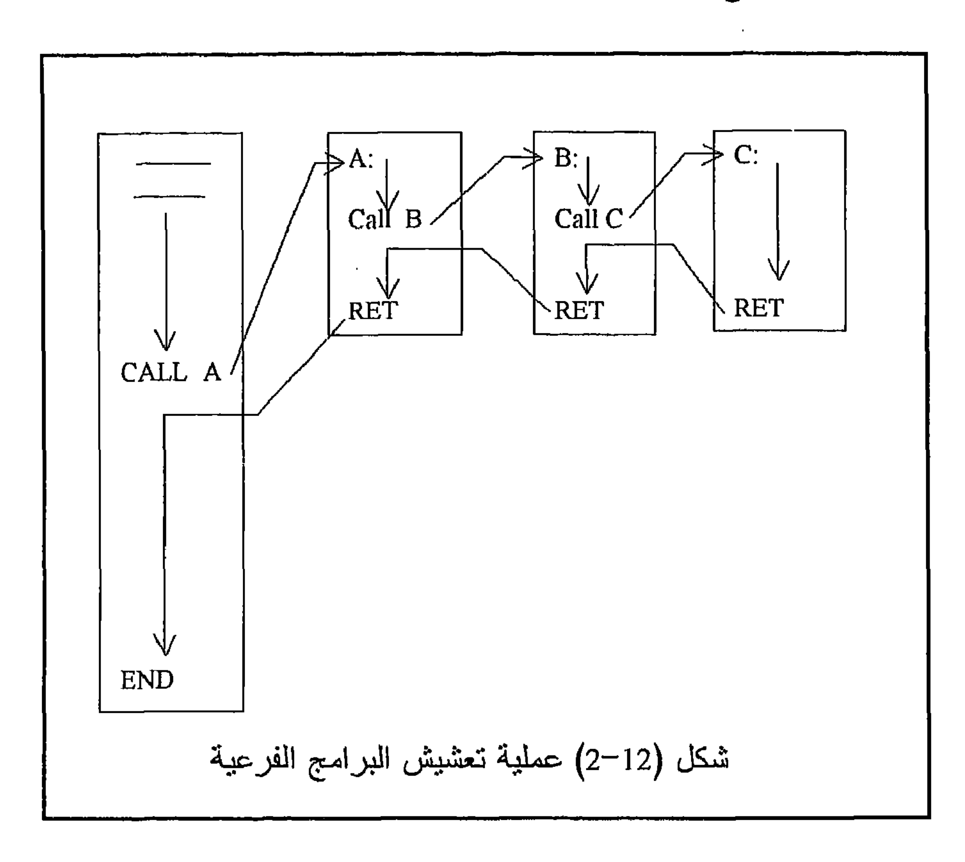
التداخلات . هذه العملية تسمى عملية تعشيش nesting للبرامج الفرعيـــة . بعـــد الانتهاء من تنفيذ آخر برنامج فرعى فى السلسلة وليكن البرنـــامج الفرعـــى (ج) فإن المعالج يرجع إلى البرنامج الفرعى (ب) من حيث تم النداء علـــى البرنـــامج الفرعى (ج) وتتم تكملة البرنامج الفرعى (ب) حيث يرجع المعالج إلى البرنـــامج الفرعى (أ) من حيث تم النداء على البرنامج الفرعى (ب) ، بعد الانتهاء من تنفيذ البرنامج الفرعى (أ) تتم العودة إلى البرنامج الأساسى من حيث تم النـــداء علـــى البرنامج الفرعى (أ) .



شكل (12-1) البرنامج الفرعى والبرنامج الأساسى

يجب أن نحذر خطأ أو فخا يمكن أن نقع فيه وهو أن ينادى واحد من الـــبرامج الفرعية اللاحقة أحد البرامج الفرعية السابقة كأن ينـــادى مثـــلا البرنـــامج (ج) البرنامج (ب) أو (أ) . فى هذه الحالة سيدور المعالج فى حلقة لانهائية لا مخـــرج منها ولن يرجع المعالج أبدا إلى البرنامج الأساسى الذى خرج منه حيث ســـيظل البرنامج (ج) ينادى على (ب) والبرنامج (ب) ينادى على (ج) إلى ما لا نهاية .
هناك الكثير من أوامر النداء على والعودة من البرامج الفرعية . فمنها مـــا هـــو غير مشروط مثل الأمر CALL addr للشـــريحتين 8085 و Z80 بحيـــث ينتقـــل التنفيذ إلى العنوان addr دون أى شرط مثله فى ذلك مثل الأمـــر JMP . وفـــى

المقابل هناك أمر العودة RET للمعالج 8085 و Z80 الذى يكون آخر أمـــر فـــى البرنامج الفرعى والذى عند تنفيذه تتم العودة دون أى شرط إلى المكان الذى تـــم النداء منه . هناك أيضا النداء المشروط على البرامج الفرعية والعودة المشـــروطة من البرامج الفرعية مثل الأمر CZ addr للمعالج 8085 والذى يعنى نداء علــــى برنامج فرعى مشروط بعلم الصفر يساوى واحدا . كما أن هناك أيضـــا العـــودة المشروطة بعلم الصفر يساوى واحدا وهو الأمـــر RZ للمعـــالج 8085 . راجـــع الفصل الخاص ببرمجة المعالج الذى تستخدمه للتعرف على جميع أوامر النـــداء والعودة من البرامج الفرعية .



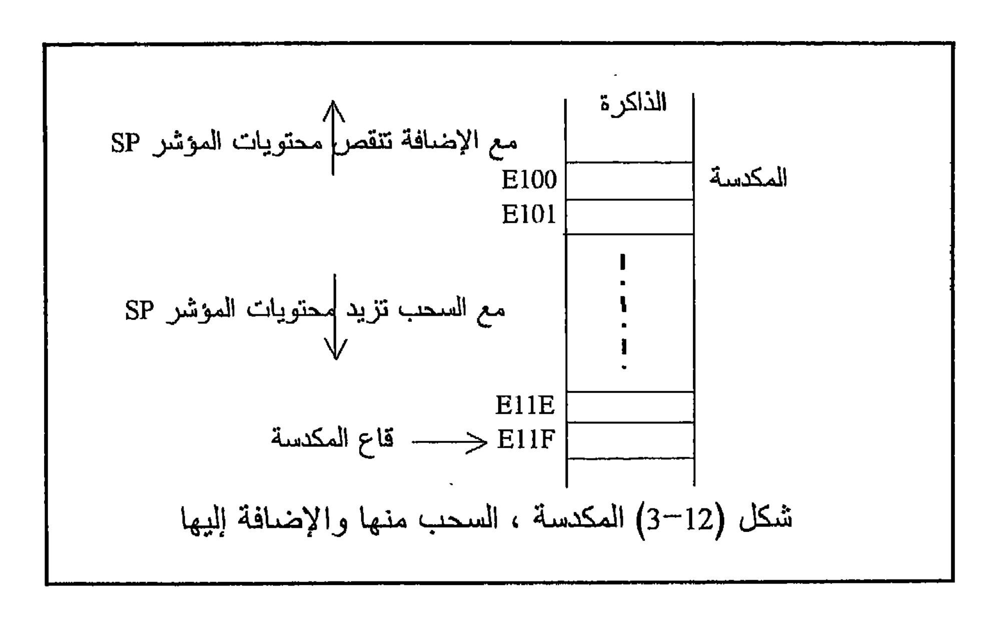
شكل (12-2) عملية تعشيش البرامج الفرعية

12-3 كيف يعود المعالج إلى نفس المكان الذى خرج منه ؟

إن السر يكمن فى المكدسة stack ومؤشر المكدسة stack pointer . المكدسة هـــى جزء مقتطع من الذاكر RAM الملحقة على المعالج لخدمة أغراض النداء والعودة من البرامج الفرعية وأيضا لخدمة أغراض المقاطعة وأغراض أخرى كما سنرى

فى فصل آخر . إن أقرب تشبيه للمكدسة هو الجوال (الشوال) الذى نضيف إليه من فوهته وعندما نأخذ منه فإننا نأخذ من فوهته أيضا ، أى أن آخر ما وضعنا فى الشوال (المكدسة) يكون أول ما نأخذ منها أو Last In First Out وتختصر LIFO . تشبيه آخر للمكدسة هو رص الأطباق ، فأنت حينما ترص الأطباق رأسيا تضيف إلى قمة المكدسة وعندما تريد أخذ طبق فإنك تأخذ آخر طبق وضعته على القمة .

وأما مؤشر المكدسة stack pointer, SP فإنه مسجل مكون من 16 بت محتوياته هى عنوان قمة أو آخر مكان تم التخزين فيه فى المكدسة . عندما تكون المكدسة فارغة فإن مؤشر المكدسة يشير إلى قاعها وتكون محتويات ال SP هى عنوان آخر مكان فى المكدسة . عند الإضافة إلى المكدسة فإن المؤشر ينقص محتوياته وعند السحب من المكدسة فإن المؤشر تزيد محتوياته بحيث أن الزيادة أو النقص تكون دائما بمقدار 2 لكل عملية سحب أو إضافة كما فى شكل (12-3) .



شكل (12-3) المكدسة ، السحب منها والإضافة إليها

إن الإضافة والسحب من المكدسة تكون دائما على أزواج المسجلات ولا يمكنك بأى حال أن تضيف أو تسحب مسجلا واحدا فقط أو بايت واحدة فقط من أو إلى المكدسة . لذلك فإن مؤشر المكدسة حينما يزيد أو ينقص فإنه يزيد أو ينقص بمقدار اثنين ، أى اثنين بايت ، ولا يمكن أن يزيد أو ينقص بمقدار واحد على الإطلاق . شكل (12-4) يبين الخطوات التى يقوم بها المعالج عند تنفيذ أمر النداء على أى برنامج فرعى وهى كالتالى :

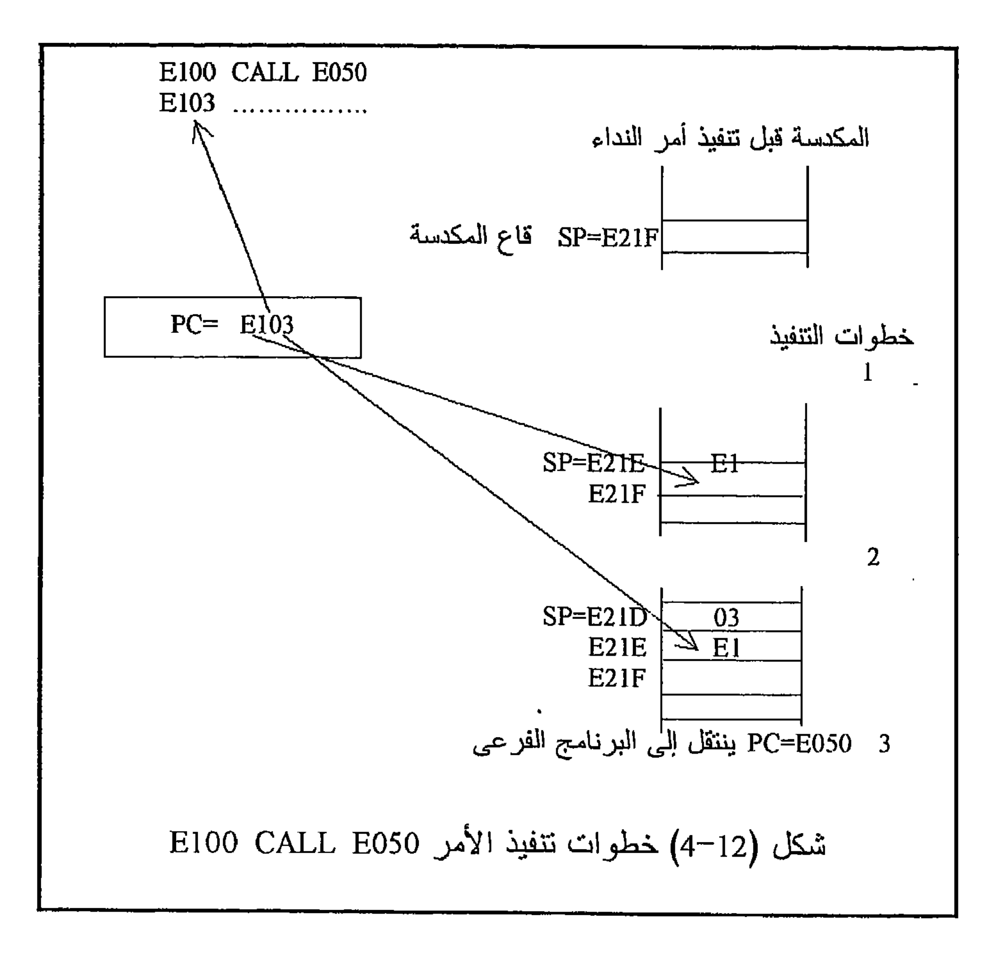
1. مؤشر المكدسة SP ينقص decrement بمقدار واحد ، ويخزن البايت ذات

القيمة العظمى من عداد البرنامج PC فى هذا العنوان .

2. مؤشر المكدسة SP ينقص بمقدار واحد آخر وتخــــزن البـــايت ذات القيمـــة الصغرى من عداد البرنامج فى العنوان الجديد . بذلك يكون قد تم الحفاظ علـــى محتويات عداد البرنامج وهى عنوان الأمر الذى عليه الدور فى التنفيذ بعد الأمر CALL فى المكدسة .

3. يحمل عداد البرنامج بعنوان أول بايت فى البرنامج الفرعى وهـــو العنـــوان الموجود فى أمر النداء CALL addr . بذلك ينتقل التنفيذ إلى البرنامج الفرعى .

لاحظ من شكل (12-4) أن قاع المكدسة وهى العنوان E21F تكون فارغة دائمـــا وذلك لأن عملية إنقاص مؤشر المكدسة تكون دائما قبل التخزين فيها . فى نهايـــة تنفيذ البرنامج الفرعى يكون آخر أمر ينفذ هو الأمر RET وعند تنفيذ هذا الأمـــر تتم العمليات العكسية للعمليات الموضحة فى شكل (12-4) وهى كالتالى :



شكل (12-4) خطوات تنفيذ الأمر E100 CALL E050

1. محتويات قمة المكدسة فى العنوان E21D وهى الرقم 03 تدفع إلـــى البـــايت

ذات القيمة الصغرى من عداد البرنامج وتزداد قيمة محتويات مؤشر المكدسة بمقدار واحد فيصبح E21E .

2. محتويات القمة الجديدة للمكدسة فى العنوان E21E وهى الرقم E1 تدفع إلى البايت ذات القيمة العظمى من عداد البرنامج ، وتزداد قيمة محتويات مؤشر المكدسة بمقدار واحد آخر فتصبح E21F وهى قمة المكدسة (أو قاعها) التى كانت موجودة فى بداية تنفيذ الأمر CALL E050 .

3. بذلك تصبح محتويات عداد البرنامج هى عنوان الأمر الموجود عند المكان E103 وهو الأمر الذى عليه الدور فى التنفيذ فى البرنامج الأساسى بعد أمر النداء مباشرة . إن هذه الخطوات قد تختلف اختلافا بسيطا من معالج لآخر ولكن يظل المعنى الأصلى كما هو .

فى حالة البرامج الفرعية المعششة مع بعضها nested مهما كانت درجة تعشيشها فإنه عند كل أمر نداء CALL يتم تخزين عنوان الأمر التالى للأمر CALL مباشرة (أى الأمر الذى عليه الدور فى التنفيذ) وهكذا إلى أن نصل إلى آخر برنامج فرعى فى السلسلة حيث سيكون الأمر RET فى آخره هو أول أمر RET يتم تنفيذه ونتيجة له يحمل عداد البرنامج بأول اثنين بايت من قمة المكدسة فيرجع التنفيذ إلى البرنامج الفرعى قبل الأخير وهكذا مع كل أمر RET يسحب عنوان (2 بايت) من قمة المكدسة ويرجع التنفيذ إلى برنامج فرعى سابق إلى أن يصل التنفيذ إلى حيث انتهى من البرنامج الأساسى .

هناك أمر يمكنك من تخزين أى زوج مسجلات فى المكدسة وهو الأمر :

PUSH rp

الذى يقوم بتنفيذ أول خطوتين فى شكل (12-4) ، مع ملاحظة أن محتويات عداد البرنامج لا تتغير هنا على الإطلاق لأنه ليس هناك أى قفز . كذلك فإن الأمر :

POP rp

يقوم بالعملية العكسية للأمر PUSH ، أى يسحب اثنين بايت من المكدسة ويضعهم فى زوج المسجلات المذكور فى الأمر POP وينقص محتويات مؤشر المكدسة بمقدار اثنين . هذه العملية تكون مهمة جدا عند خدمة المقاطعة كما سنرى فى فصل قادم وهذان الأمران موجودان لجميع المعالجات التى ندرسها ويمكن مراجعتهما فى قوائم الأوامر .

إن محتويات مؤشر المكدسة التى تشير إلى قاعها أى أول مكان فيها يتم تحديدها عن طريق المبرمج فى البرنامج باستخدام الأمر LXI SP,addr حيث يقوم هذا الأمر بتحميل المسجل SP بالعنوان addr الذى يختاره المبرمج ليكون عنوان قاع المكدسة وذلك فى المعالج 8085 ، أو LD SP,addr فى حالة المعالج Z80 (لاحظ أنه قبل أى تعامل مع المكدسة تكون قمتها هى قاعها أى وهى فارغة) .

بذلك نكون قد أنهينا أهم ما يتعلق بالمكدسة ومؤشر المكدسة والبرامج الفرعية وفوائدها وننتقل الآن إلى بعض الأمثلة كتطبيقات على ذلك . لاحظ أن كل ما تم شرحه فى هذا الباب يمكن تطبيقه على أى معالج من المعالجات التى درسناها (والتى سندرسها بخلاف طفيف) ، فقط نكون حذرين عند كتابة شكل الأمر بالأسمبلى أو شفرته الستعشرية ، ولكن الأساس أو الهيكل العام لفكرة المكدسة والبرامج الفرعية التى شرحناها هى نفسها دائما .

4-12 حساب أزمنة التأخير

مثال 12-1

لدينا بوابة الإخراج رقم 00 وموصلا عليها ثمانية دايودات أو موحدات ضوئية كل دايود موصل على بت من بتات البوابة . المطلوب إنارة هذه الدايودات بالتتابع بحيث أن الدايود الأول يضىء وبعده بنصف ثانية يضىء الدايود الثانى وبعده بنصف ثانية أخرى يضىء الدايود الثالث وهكذا حتى تصبح كل الدايودات مضيئة ، ثم بعد آخر دايود بنصف ثانية تطفأ جميع الدايودات ثم يبدأ فى الإضاءة من جديد بنفس الطريقة السابقة . اكتب برنامجا يقوم بهذه المهمة مستخدما البرامج الفرعية .

تعتمد فكرة البرنامج على عمل برنامج فرعى للتأخير بزمن مقداره نصف ثانية ثم ننادى على هذا البرنامج من البرنامج الأساسى بعد إنارة كل دايود . قبل أن ندخل فى تفاصيل البرنامج نريد أن نوضح هنا كيفية الحصول على أى زمن تأخير بأى قيمة . تقوم الفكرة أساسا على عمل حلقة ينفذها المعالج عددا من المرات دون التأثير على أى شىء فى البرنامج وبمعلومية عدد مرات تنفيذ هذه الحلقة وعدد الأوامر فيها وزمن تنفيذ كل أمر نستطيع حساب الزمن الكلى لتنفيذ الحلقة . انظر مثلا لهذا البرنامج البسيط :

```
      MVI C,FF;   7
LOOP: DCR C;      4
      JNZ LOOP;  10
```

الرقم الذى على يمين كل أمر هو عدد نبضات التزامن clock التى يأخذها الأمر حتى يتم تنفيذه وهذا العدد موضح فى قوائم أوامر كل واحد من المعالجات التى شرحنا طريقة برمجتها فى الفصول السابقة . فإذا كان تردد التزامن clock المستخدم هو 2 ميجاهرتز فإن ذلك يعنى أن زمن كل نبضة هو نصف ميكروثانية . لذلك فإن زمن تنفيذ الأمر الأول مثلا سيكون 3.5 ميكروثانية والأمر الثانى سينفذ فى 2 ميكروثانية والثالث سينفذ فى 5 ميكروثانية وهكذا . فى البرنامج السابق نلاحظ أن زمن تنفيذ الحلقة مرة واحدة سيكون (4 +

$(10\times0.5=7)$ ميكروثانية ، وهذه الحلقة ستنفذ عددا من المرات مقـداره 256 (FF) مرة لذلك فإن مثل هذه الحلقة ستعطى زمـــن تـــأخير مقـداره $256\times7=1792$ ميكروثانية ، وأما البرنامج الســـابق كلــه فســيعطى $1792 + 3.5 = 1795.5$ ميكروثانية . إذا كان هذا الزمن يكفى كتأخير لما تحتاج فقد انتهيت وإلا فعليـــك البحث عن طريقة تطيل بها هذا الزمن ، ومن ذلك استخدام زوج من المســجلات بدلا من مسجل واحد كما فى البرنامج التالى :

```
      LXI D,FFFF;  10
LOOP: DCX D;        6
      MOV A,D;      4
      ORA E;        4
      JNZ LOOP;    10
```

هذا البرنامج يقوم بتحميل الزوج DE بالقيمة FFFF(65536) ثم يدخل فى حلقـــة إنقاص بمقدار واحد إلى أن تصل محتويات زوج المسجلات إلى أصفار وعندهـــا تنتهى الحلقة . لاحظ أن الأمر DCX ليس له تأثير على الأعلام لذلك فقد تم نقــل محتويات مسجل من الاثنين إلى المركم ثم نفذت عملية OR على المسجلين والتى لن تكون نتيجتها صفرا إلا إذا كانت محتويات كل من المسجلين أصفارا . زمـــن التأخير الذى سيعطيه هذا البرنامج سيكون :

$24\times0.5\times65536+10\times0.5=786437$ ميكروثانية .

$=0.79$ ثانية تقريبا .

إذا كان هذا الزمن يفى بما تحتاج إليه من زمن تأخير فقد انتهيت ، وإلا فعليـــك استخدام الحلقات المعشقة كما فى البرنامج التالى :

```
       MVI B,FF;     7
LOOP1: MVI C,FF;     7
LOOP2: DCR C;        4
       JNZ LOOP2;   10
       DCR B;        4
       JNZ LOOP1;   10
```

زمن تأخير الحلقة الداخلية = $14\times0.5\times256$

= 1792 ميكروثانية تقريبا .

زمن تأخير الحلقة الخارجية= $((7+4+10)\times0.5+1792)\times256$

= 461440 ميكروثانية تقريبا .

زمن التأخير الكلى للبرنامج= $3.5 + 461440$

= 461443.5 ميكروثانية .

= 0.46 ثانية تقريبا .

لاحظ أنه وإن كان زمن التأخير الناتج من حلقتين معشقتين أقل من زمن التـــأخير الناتج من زوج من المسجلات إلا أن الحلقات المعشقة يمكن تعشيقها لأى درجـــة

فمثلا فى داخل الحلقة LOOP2 يمكن عمل حلقة ثالثة LOOP3 وهكذا للحصــول على أزمنة تأخير كبيرة . أحيانا يكون المطلوب أزمنة محددة بقدر الإمكان كمــا فى المثال الذى نحن بصدده الآن حيث المطلوب هو زمن تــأخير مقــداره 0.5 ثانية بالضبط . فى هذه الحالة يمكن استخدام الأمر NOP فى أماكن معينــة فــى حلقات التأخير للضبط الدقيق للأزمنة المطلوبة . مثلا ماذا سيكون الوضــع لــو أضفنا الأمر NOP داخل الحلقة الداخلية LOOP2 فى البرنامج السابق ؟ وإجابــة على ذلك فمن المعروف أن الأمر NOP يأخذ 4 نبضات تزامن لذلــك ســيصبح زمن التأخير الجديد هو 0.59 ثانية تقريبا ، وهذه القيمة أبعد من القيمة المطلوبــة !! فى هذه الحالة يمكن وضع الأمر NOP فى الحلقة الخارجية ، وإذا لــم يــف (يفى) بالغرض المطلوب فيمكن تركه فى الحلقة الداخلية مع التغيير فــى القيمــة الابتدائية فى أى من المسجلين B أو C ، ولقد وجدنا أنه بوضع القيمة الابتدائيــة D8 فى المسجل B بدلا من القيمة FF فإن زمــن التــأخير فــى هــذه الحالــة يساوى499932 ميكروثانية وهى أقرب شيء إلى نصف الثانية . أى أن التعديــل البسيط فى القيمة النهائية لزمن التأخير ممكن بعدة طرق .

بعد أن رأينا كيفية الحصول على زمن التأخير المطلوب فإن البرنامج التالى يبيــن عملية الإضاءة التتابعية ، وهذا البرنامج مكتوبا بلغة الأسمبلى للشــريحة 8085 . هذا البرنامج كتب بصورة مبسطة جدا ويستطيع كل قارئ أن يأتى ببرنامج آخــر يؤدى نفس الهدف وقد يكون أبسط من ذلك . ولقد تعمدنا كتابة البرنامج باستخدام العلامات LABELS حتى نترك للقارئ حرية كتابة البرنامج فى أى مكــان فــى الذاكرة يريده . بذلك نكون قد انتهينا من إعطاء القارئ فكرة كافية عن الــبرامج الفرعية وما يتعلق بها من المكدسة ومؤشر المكدسة .

```
START:  MVI A,01
        OUT 00
        CALL DELAY
        MVI A,03
        OUT 00
        CALL DELAY
        MVI A,07
        OUT 00
        CALL DELAY
        MVI A,0F
        OUT 00
        CALL DELAY
        MVI A,1F
        OUT 00
        CALL DELAY
        MVI A,3F
```

```
        OUT 00
        CALL DELAY
        MVI A,7F
        OUT 00
        CALL DELAY
        MVI A,FF
        OUT 00
        CALL DELAY
        MVI A,00
        OUT 00
        CALL DELAY
        JMP START
DELAY:  MVI B,DB
LOOP1:  MVI C,FF
LOOP2:  DCR C
        NOP
        JNZ LOOP2
        DCR B
        INZ LOOP1
        RET
```

5-12 تـمـاريـن

1. شرح دور المكدسة مع البرامج الفرعية ؟
2. ما هو الفرق بين القفز العادى فى أى برنامج والقفز إلى برنامج فرعى ؟
3. اشرح دور الأمر RET (أمر العودة من البرامج الفرعية) فى ضمان عـودة المعالج إلى نفس المكان الذى خرج منه فى البرنامج الأساسى ؟
4. لماذا يكون من الضرورى عادة استخدام الأمر PUSH فـى أول البرنـامج الفرعى والأمرPOP فى آخره ؟
5. يحتوى هذا الفصل على بعض البرامج الفرعية للحصول على أزمنة التـأخير ، فهل يتغير مقدار هذه الأزمنة بتغير التزامن clock المستخدمة ؟
6. اكتب برنامجا فرعيا للحصول على زمن تأخير مقداره 100ميللثانية مع العلـم أن التزامن يساوى 2 ميجاهرتز؟
7. اكتب برنامجا فرعيا آخر يستخدم البرنامج الفرعى السابق للحصـول علـى زمن تأخير مقداره ثانية واحدة ؟
8. اكتب برنامجا فرعيا آخر يستخدم البرنامجين السابقين للحصول على زمـن تأخير مقداره دقيقة واحدة ؟

9. استخدم البرامج الفرعية السابقة فى عمل ساعة رقمية تظهر الثوانى والدقائق والساعات على مظهرات السبع قطع 7 segment ؟

10. اكتب برنامج التحكم فى إشارات المرور فى الفصل السابق مستخدما الـبرامج الفرعية ولاحظ الفرق ؟

11. اكتب برنامج يحسب قيمة X التالية :

$$X= 5! + 8! + 9!$$

حيث ! تمثل مضروب الرقم .

12. اكتب برنامج يحسب قيمة X التالية :

$$X= 5^5 + 8^4 + 9^3$$

الفصل الثالث عشر

المقاطعة

Interrupt

13-1 مقدمة

لقد رأينا فى الفصول السابقة كيفية مواجهة المعالج مع الأجهزة المحيطة سواء كانت ذاكرة أو بوابات إدخال وإخراج . فى العادة يقوم المبرمج بكتابة برنامج على أساسه يقوم المعالج بخدمة هذه الأجهزة المحيطة . هناك طريقتان يمكن للمعالج أن يخدم بهما هذه الأجهزة المحيطة وهما :

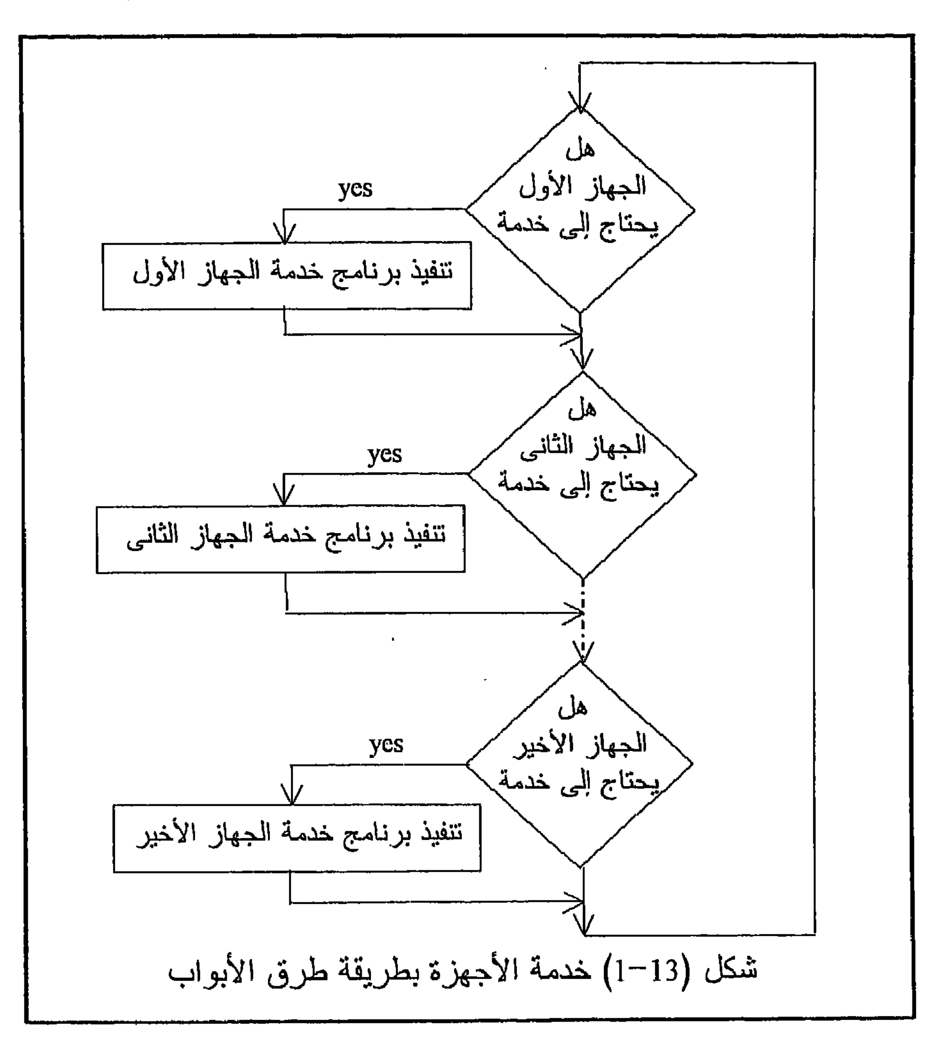
1. الطريقة الأولى سنسميها "خدمة طرق الأبواب" التى عرفت فى المراجع الإنجليزية بكلمة polling والتى لها نفس المعنى كما سنرى ولكن الترجمة ليست حرفية .
2. الطريقة الثانية هى طريقة المقاطعة interrupt وهى الطريقة التى سنشرحها بالتفصيل فى هذا الفصل ، ولكن لا مانع من القاء نظرة سريعة على الطريقة الأولى حتى يختار القارىء أى الطريقتين يحب أن يستخدم .

13-2 طريقة طرق الأبواب لخدمة الأجهزة المحيطة Polling service

تعتمد هذه الطريقة على أن المعالج يقوم بطرق أبواب جميع الأجهزة المحيطة بالتتابع ويسأل كل منها هل تريد خدمة أقوم بأدائها ؟ فإذا كانت إجابة الجهاز بنعم فإن المعالج يقوم بتنفيذ هذه الخدمة له وفى الحال دون أى إنتظار ، أما إذا كانت إجابة الجهاز بالنفى فإن المعالج ينتقل إلى الجهاز التالى له ويسأله نفس السؤال ، وهكذا إلى أن يصل المعالج إلى آخر جهاز فإما أن يخدمه أو لا على حسب إجابته . بعد آخر جهاز يرجع المعالج إلى أول جهاز ويعيد الكرة من جديد إلى مالانهاية . أى أن الوظيفة الوحيدة للمعالج فى هذه الحالة هى الدوران إلى مالانهاية على الأجهزة المحيطة وتقديم الخدمة لمن يجيبه . شكل (13-1) يبين مخطط سير لوظيفة المعالج فى هذه الحالة . لقد استخدمنا هذه الطريقة إلى حد ما فى مثال إشارات المرور الذى سبق شرحه فى الفصل الثانى عشر حيث كان المعالج دائما يسأل بوابة الإدخال ، أى يقرأها ، فإذا كانت واحدا يقوم بتنفيذ برنامج معين ، وإذا كانت صفرا يقوم بتنفيذ برنامج آخر وبعد الانتهاء من أى من البرنامجين يرجع ويقرأ بوابة الإدخال مرة ثانية وهكذا إلى مالانهاية .

إن من مميزات طريقة طرق الأبواب لخدمة الأجهزة المحيطة أنها سهلة البرمجة ولا تحتاج إلى الكثير من التجهيزات hardwar . وإن من عيوبها أن المعالج يكون مخصصا لوظيفة خدمة هذه الأجهزة ولا يستطيع الفكاك منها وأنت كمبرمج لا تستطيع عادة الاستفادة منه فى أى أغراض أخرى ، وبالذات إذا كان عدد

الأجهزة التى يقوم المعالج بالمرور عليها قليلا فإن ذلك يعتبر إهدارا لفعالية المعالج . أما إذا كان عدد الأجهزة التى يقوم المعالج بخدمتها بهذه الطريقة كبيرا فإن المعالج قد يتأخر على بعض الأجهزة التى تحتاج لخدمته على فترات متقاربة لأن عليها الإنتظار لحين أن يجيء دورها ، كما أنها ليس من حقها أن تقاطع المعالج وتطلب الخدمة الفورية فى حالات الضرورة ، وكما نعلم فإن هناك الكثير من المواقف التى تستلزم الخدمة من المعالج فورا كضرب جرس إنذار مثلا عند حدوث أى طارىء فى أى نظام تحكم مثل إنقطاع الكهرباء أو حريق أو إرتفاع زائد فى ضغط غاز وغيرها الكثير . لذلك فإن المعالج يقدم للمبرمج نوعا آخر من الخدمة وهى المقاطعة التى سنتكلم عنها بالتفصيل فى الجزء القادم .



شكل (13-1) خدمة الأجهزة بطريقة طرق الأبواب

13-3 المقاطعة Interrupt

فى نظام خدمة الأجهزة الخارجية عن طريق المقاطعة لا يذهب المعالج إلى الأجهزة ويطرق بابها ليعرض عليها خدماته فإن أرادت أعطى وإن أبت يذهب إلى جهاز آخر ، لا ، بل إن المعالج هنا يكون عادة مشغولا فى تنفيذ برنامج معين وعادة ما يكون هذا البرنامج لا نهائى فإذا احتاج أحد الأجهزة لخدمة من المعالج فإنه يقاطعه ويطلب منه الخدمة فيقوم المعالج بتنفيذ هذه الخدمة للجهاز المقاطع وبعد الانتهاء من هذه الخدمة يعود المعالج لتنفيذ البرنامج الأساسى من حيث انتهى قبل المقاطعة . إن ذلك تماما مثلما أنك تكون واقفا مع أحد زملائك تتحدثان ، ثم يجىء آخر ويقاطعكما ليسأل عن موعد محاضرة المعالجات ، فى هذه الحالة يقطع المتكلم محادثته ليجيب السائل عن طلبه ثم بعد الإجابة يستأنف حديثه الأول . إن البرنامج الذى يقوم المعالج بتنفيذه عند المقاطعة يسمى برنامج خدمة المقاطعة .

من مميزات هذه الطريقة أن الأجهزة المقاطعة تستطيع مقاطعة المعالج فى أى وقت تريد وليس عليها الإنتظار إلى دورها مثل الطريقة السابقة ، وإذا تصادف وتمت المقاطعة فى نفس الوقت من أكثر من جهاز فإن المعالج يخدمها حسب أولويات تحدد له من قبل المستخدم مسبقا ، فكيف يتم ذلك ؟ فى أثناء انشغال المعالج فى تنفيذ برنامج خدمة مقاطعة معينة ، هل يستطيع جهاز آخر أن يقاطعه؟ كيف نكتب برنامج خدمة المقاطعة ؟ وكيف يرجع المعالج إلى نفس مكانه فى البرنامج الأساسى بعد إنتهاء خدمة المقاطعة ؟ كل هذه الأسئلة وأكثر سنجيب عليها من خلال دراستنا للأجزاء القادمة وبعد دراستنا لتفاصيل مقاطعة كل معالج من المعالجات التى درسناها .

إن المقاطعة تكون عادة عن طريق إشارة يرسلها الجهاز المقاطع إلى المعالج على أحد أطرافه المخصصة لذلك وعندما يكتشف المعالج هذه الإشارة فإنه يقوم بتنفيذ برنامج خدمة المقاطعة وذلك يرجع لأن المعالج يقوم بقراءة جميع أطراف المقاطعة وإختبارها قبل الدخول فى تنفيذ أى أمر . وإليك بعض الأمثلة التى تستخدم فيها المقاطعة :

- الأجهزة الخارجية مثل الطابعة ولوحة المفاتيح يمكنها أن تقاطع المعالج وترسل أو تستقبل أى معلومات .
- يمكن فى أى وقت مقاطعة أى برنامج يتم تنفيذه إذا كان هذا البرنامج ينفذ بطريقة خطأ .
- يمكن للعمليات الصناعية التى يتم مراقبتها باستخدام المعالج أن تقاطعة فى أى لحظة طوارىء تحدث للعملية الصناعية .

عند إعطاء إشارة المقاطعة لأى معالج من المعالجات التى نقوم بدراستها يحـــدث الآتى مع بعض الإختلافات البسيطة من معالج لآخر سنشير إليها فى موضعها .

1. الأمر الحالى يتم إكمال تنفيذه من قبل المعالج .
2. عنوان الأمر الذى عليه الدور فى التنفيذ (محتويات عداد البرنامج) تخزن فى المكدسة Stack حتى يمكن العودة إليه عند الانتهاء من خدمة المقاطعة . كما يتـم تخزين محتويات أى مسجل يخشى من تغيير محتوياته فى أثناء خدمة المقاطعـــة ويتم ذلك عن طريق المبرمج .
3. كل إشارة مقاطعة لها عنوان خاص مصاحب لها كما سنرى ، يتم وضع هـذا العنوان فى عداد البرنامج (عن طريق المعالج) حيث يقفز المعـــالج إلـــى هـــذا العنوان ويبدأ فى تنفيذ البرنامج الذى يكون هذا هو عنوان أول أمر فيه ويســـمى هذا البرنامج ببرنامج خدمة المقاطعة وتتم كتابته عن طريق المستخدم .
4. بعد الانتهاء من برنامج خدمة المقاطعة يعود المعالج إلى البرنامج الأصلـــى ليستأنف تنفيذه من نفس مكان المقاطعة بالاستعانة بالعنوان الذى تم تخزينه فـــى المكدسة كما فى الخطوة رقم 2.

إن الجهاز المقاطع فى الكثير من الأحيان وبعد إعطاء إشارة المقاطعـــة وقبـــول المعالج لها والبدء في برنامج الخدمة فإن الباب يظل مفتوحا للأجـــهزة الأخـــرى للمقاطعة مما قد ينشأ عنه موقف يجب الحرص منه . لو أن المعالج بدأ فى خدمة المقاطعة وما زال طرف المقاطعة الموصل على الجهاز المقـــاطع فعـــالا فـــإن المعالج سيبدأ فى خدمة نفس المقاطعة من جديد على الرغم من أنه لم ينته مـــن الخدمة السابقة ، وبمجرد أن يبدأ فى خدمة المقاطعة من جديد للمرة الثانية ومـــا زال خط المقاطعة فعالا من قبل الجهاز المقاطع فإن المعالج سيبدأ فى الخدمة من جديد أيضا ، وهكذا فإن المعالج سيدخل فى حلقة لا نهائية لن يخرج منها دون أن ينفذ برنامج خدمة المقاطعة . لتجنب هذا الموقف يجب على المـــبرمج أن يضـــع أمرا معينا فى بداية برنامج خدمة المقاطعة يمنع المعالج من خدمة أى مقاطعـــة إلى أن ينتهى من الخدمة الحالية التى دخل فيها . إن هذه العملية لــــها تفـــاصيل ستختلف من معالج إلى آخر لذلك سنرجىء الكلام عنها الآن .

13-4 مقاطعة المعالج 8085

شكل (13-2) يبين جدولا لجميع أطراف المقاطعـــة الخاصـــة بالشـــريحة 8085 وعنوان المكان الذى يتم القفز إليه عند إعطاء إشارة المقاطعة على هذا الطـــرف وكذلك الأولوية الخاصة بكل طرف من أطراف المقاطعة وكيفية إعطـــاء هـــذه الإشارة على كل طرف . كما نلاحظ من هذا الشكل فإن الطرف TRAP له أعلى أولوية ثم يليه الطرف RST7.5 فالطرف RST6.5 ثم الطرف RST5.5 ثم يكـــون

الطرف INTR له أقل أولوية . إن كلمة أولوية هنا تعنى أنه إذا جاءت إشـــارتان على خطين مختلفين من خطوط المقاطعة فى نفس الوقت تماما فإن الإشارة التـــى على الخط ذى الأولوية الأعلى هى التى تؤخذ فى الاعتبار أما الإشارة الأخـــرى فتهمل .

كيفية إعطاء الإشارة على الطرف	العنوان الذى يتم القفز إليه	الأولوية	طرف المقاطعة
الحافة صفر إلى واحد ويبقى واحد الى أن تقبل المقاطعة	0024H	1	TRAP
الحافة صفر إلى واحد حيث يتـــم مسك هذا الواحد فى المعالج	003CH	2	RST7.5
يبقى واحد إلى أن تقبل المقاطعة	0034H	3	RST6.5
يبقى واحد إلى أن تقبل المقاطعة	002CH	4	RST5.5
يبقى واحد إلى أن تقبل المقاطعة	له تفاصيل	5	INTR

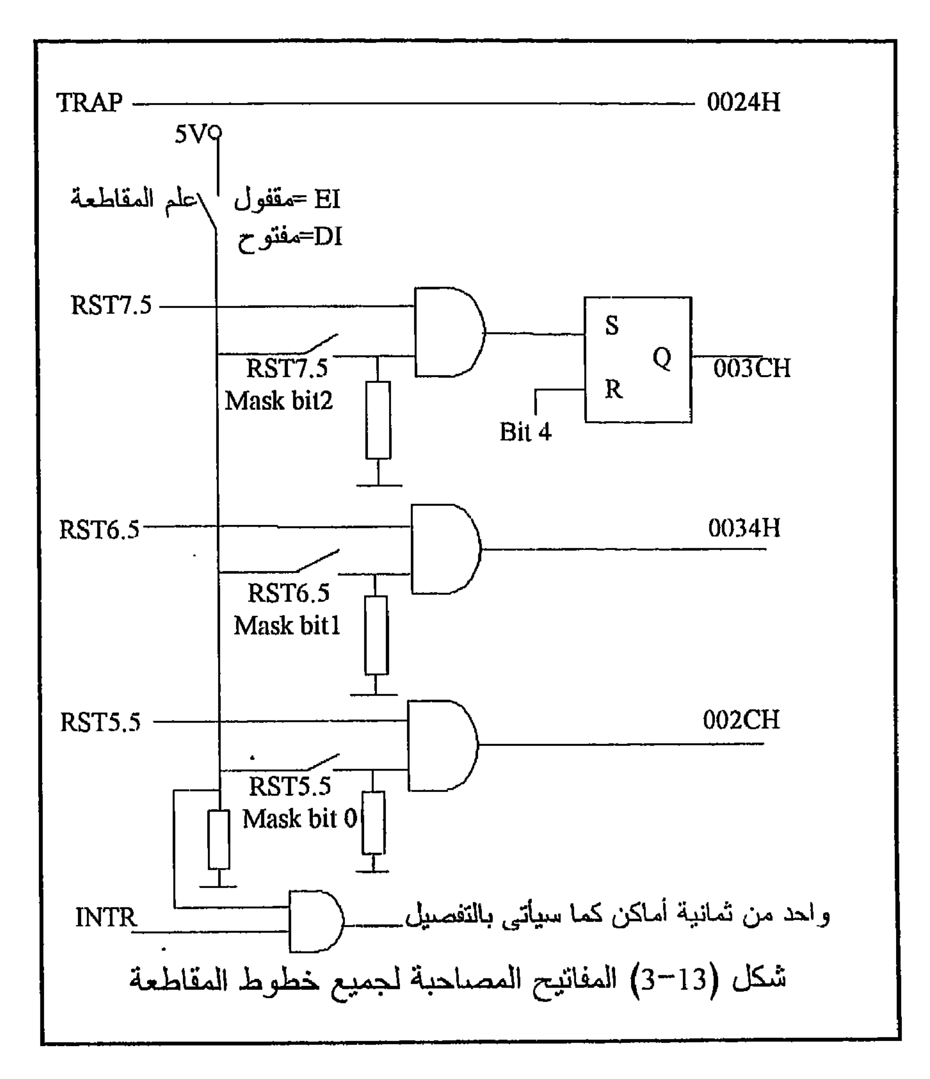
شكل (13-2) أطراف المقاطعة للشريحة 8085 وأولوياتها وعناوين القفز الخاصة بكل منها وكيفية إعطاء كل إشارة على هذه الأطراف

13-4-1 الخطوط RST5.5, RST6.5, RST7.5

بالنسبة للمعالج8085 هناك بعض الأوامر المتعلقة بالمقاطعة والتى يجب معرفتـــها أولا ومن هذه الأوامر ما يلى :

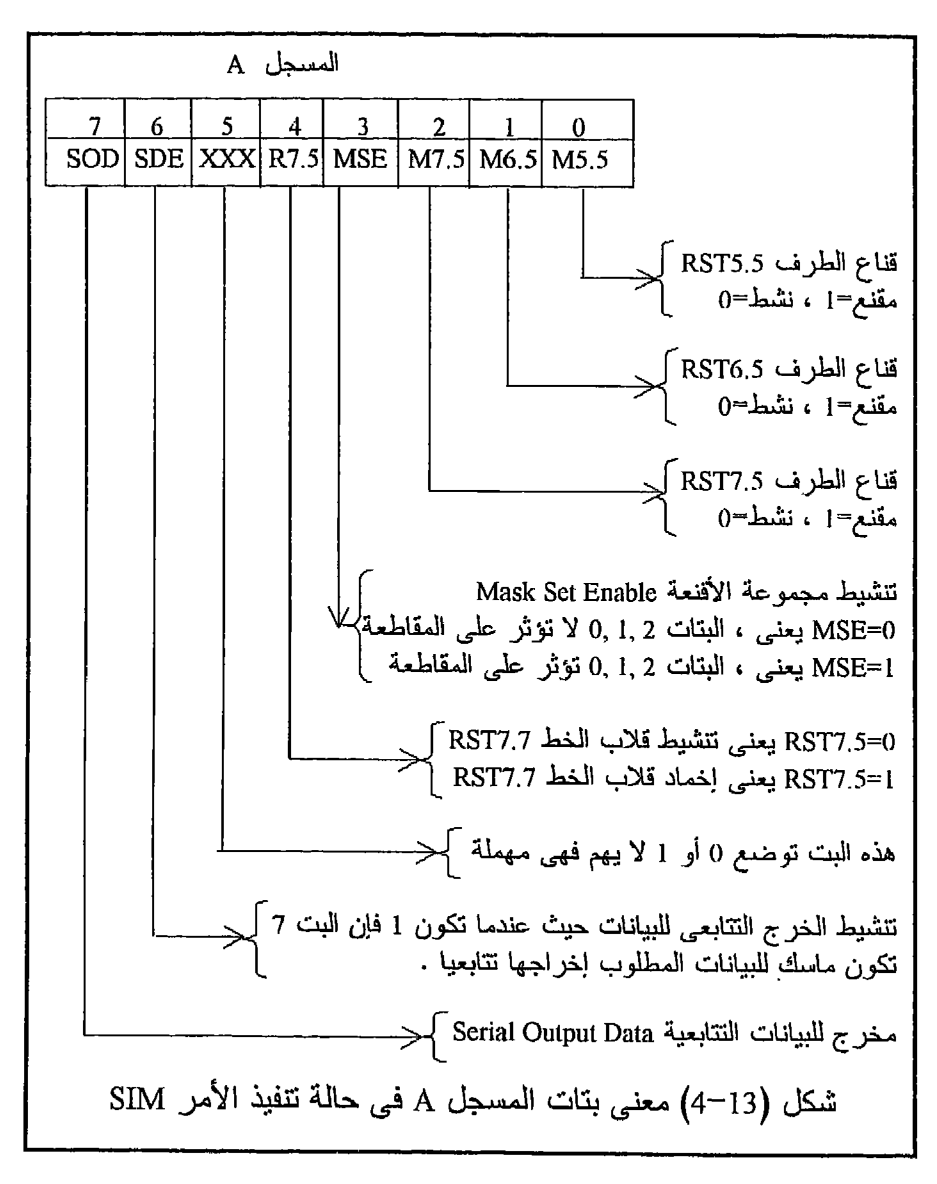
- الأمر EI ومعناه تنشيط المقاطعة Enable Interrupt . هناك فى داخل شـريحة المعالج قلابا flip flop يسمى قلاب تنشيط المقاطعة ، Interrupt Enable flip flop أى أن أى إشارة مقاطعة على جميع الخطـــوط (RST5.5, RST6.5, RST7.5, INTR) ما عدا الخط TRAP لن تنفذ إلا إذا كان خرج هذا القلاب يساوى واحـدا. لذلك فإن هذا القلاب يعتبر بمثابة مفتاح مركب على التوالى مع هـــذه الخطـــوط مجتمعه يتم فتحه (ON) بالأمر EI . شكل (13-3) يبين وضع هذا المفتاح بالنسبة لهذه الخطوط .
- الأمر DI ومعناه إخمد أو امنع أو لا تقبل المقاطعة Interrupt Dissable وكما نرى فإن تأثيره على قلاب تنشيط المقاطعة يكون تماما عكس تأثير الأمـــر EI . يستخدم الأمر DI فى حجب أو إخماد المقاطعة فى بعض أجزاء من البرنامج ومن أهمها برامج خدمة المقاطعة نفسها . لذلك فإنه يجب على المبرمج أن يضع الأمر DI فى بداية أى برنامج لخدمة المقاطعة لحماية هذه المقاطعة من نفســـها ومـــن الدخول فى الحلقة اللانهائية التى ذكرناها منذ قليل . فى نهايـــة برنـــامج خدمـــة المقاطعة يضع المبرمج الأمر EI لتنشيط المقاطعة من جديد .

• مثلما أن هناك مفتاحا عموميا (علم المقاطعة) لجميع أطراف المقاطعة ما عدا الخط TRAP يمكن تنشيطها به أو حجبها به عن طريق الأمرين EI و DI فإن هناك لكل واحد من الخطوط RST7.5 و RST6.5 و RST5.5 مفتاحا خاصا به يمكن به تنشيط هذا الخط أو حجبه ، كما أن الخط RST7.5 له مفتاح أو قلاب آخر إضافى خاص به هو فقط يمكن منه حجب المقاطعة عليه حتى لو مرت من خلال المفتاح الأول وهذه ميزة للخط RST7.5 عن باقى الخطوط الأخرى . انظر لهذه القلابات أو المفاتيح فى شكل (3-13) . هذه المفاتيح يمكن تنشيطها أو حجبها عن طريق الأمر SIM والذى يعنى "ضع أقنعة المقاطعة" Set Interrupt Masks .



شكل (3-13) المفاتيح المصاحبة لجميع خطوط المقاطعة

إن هذا الأمر عند تنفيذه يتم تنشيط أو حجب أو وضع قناع على أى خط من هـذه الخطوط على ضوء شفرة توضع فى مسجل التراكم قبل هذا الأمـــر مباشـــرة . شكل (13-4) يبين علاقة بتات مسجل التراكم مع هذه المفاتيح . نلاحظ من هـذا الشكل مثلا أن قناع الخط RST5.5 هو البت رقم 0 فإذا كانت هذه البت تســـاوى صفرا فإن الخط RST5.5 يكون نشطا أى غير مقنع أو غير محجوب ، أمـــا إذا كانت هذه البت تساوى واحدا فإن ذلك يعنى أن الخط RST5.5 يكون مقنعـــا أو محجوبا .



شكل (13-4) معنى بتات المسجل A فى حالة تنفيذ الأمر SIM

نفس الشيء يمكن أن يقال عن الخطـوط RST6.5 و RST7.5 وبتاتـها المقابلـة رقمى واحد وإثنين . البت D4 تمثل المفتاح الإضافى الخاص بالخط RST7.5 فإذا كانت هذه البت تساوى واحد فإن المقاطعه RST7.5 تكون مقنعة أو محجوبـة . شكل (3-13) يبين رسما توضيحيا لمفاتيح الأقنعة لكل خط مقاطعة وكيفية التحكم بهذه المفاتيح وعلاقتها ببعضها .

مثال 13-1

اكتب الأمر الذى ينشـط المقاطعـة RST5.5 و RST7.5 ويحجـب المقاطعـة RST6.5 . طالما أن RST5.5 نشط ، إذن D0=0 وكذلك طالما أن RST7.5 نشـط إذن D2=0 وأيضا D4=0 وطالمـا أن RST6.5 مقنـع أو محجـوب إذن D1=1 وللحصول على ذلك الوضع لابد وأن تكون D3=1 . وأما D6 فلابـد أن تكـون صفرا لأننا لسنا فى حالة إخراج بيانات تتابعية وعلى ذلك فإن البت D7 لا يـهم فى هذه الحالة أن تكون صفرا أو واحدا طالما أن D6=0 . كذلك فإن D5 لا يـهم أن تكون صفرا أو واحد . وعلى ذلك فإن محتويات المركم ستكون كما يلى :

```
D7 D6 D5 D4 D3 D2 D1 D0
0  0  0  0  1  0  1  0 = 0AH
```

ولكى يتم ذلك ننفذ الأمرين التاليين :

```
MVI A,0AH
SIM
```

مثال 13-2

تخيل أن هناك جهاز إستقبال للبيانات التتابعيـة موصـلا علـى الطـرف SOD (طرف رقم 4) للشريحة 8085 ، والمطلوب هو إرسال 0 إلـى هـذا الجـهاز كتنشيط لكى يبدأ فى الإستقبال وذلك دون التأثير على حالة أطراف المقاطعة التى تم تجهيزها فى المثال الماضى .

من شكل (13. 2) نجـد أنـه حتـى نـترك أقنعـة المقاطعـة علـى البتـات D0,D1,D2,D4 كما هى يجب أن تكون D3=0 , وحتى ننشط البـت D7 كبـت للإخراج التتابعى فإن D6=1 ، ونضع D7=0 لكى نخرجه على الطـرف SOD . لاحظ أن البتات D0,D1,D2,D4 فى هذه الحالة لا يهم أن تكون صفرا أو واحـدا وسنفترضها أصفارا ، وعلى ذلك تكون محتويات المركم كالتالى :

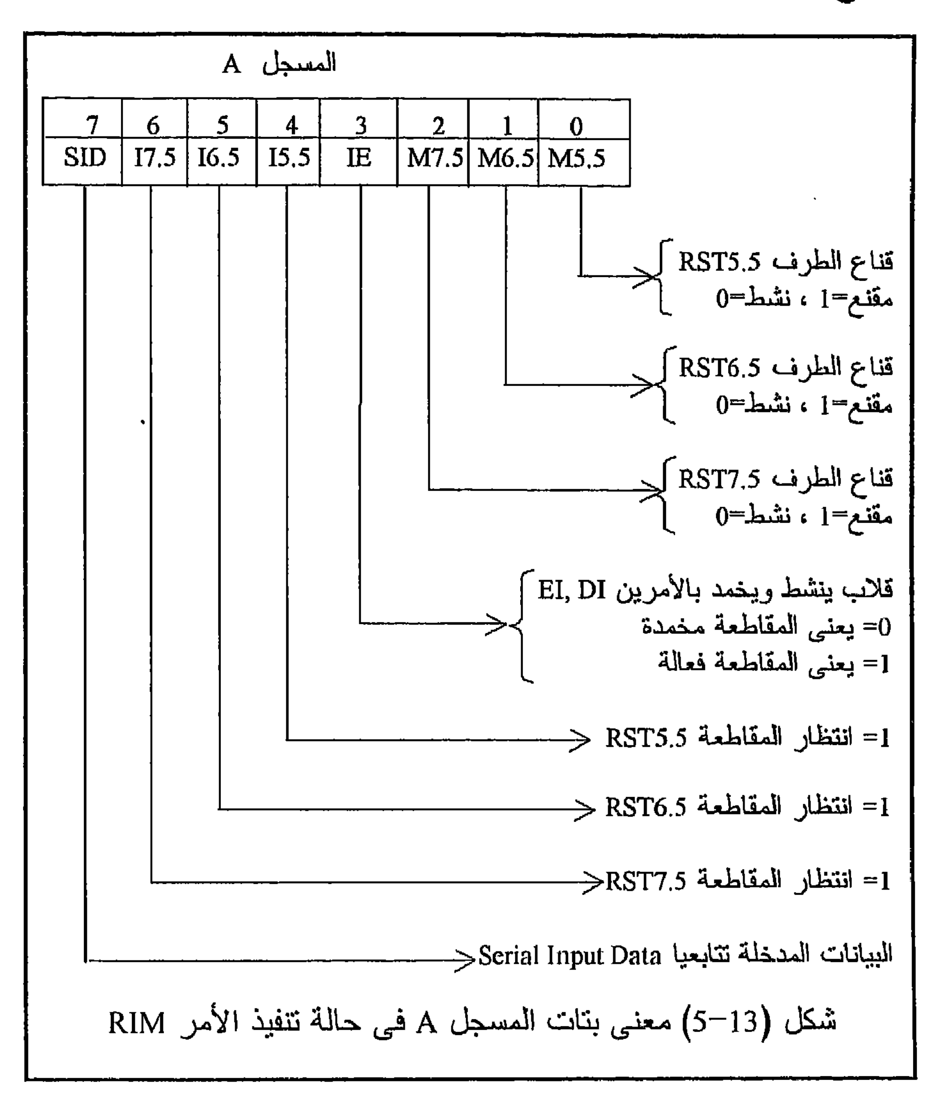
```
D7 D6 D5 D4 D3 D2 D1 D0
0  1  0  0  0  0  0  0 =40H
```

ولكى يتم ذلك ننفذ الأمرين التاليين :

```
MVI A,40H
SIM
```

لاحظ أن الخط TRAP لا يتأثر بأى شىء من ذلك .
يمكن عمل مقارنة بين خطوط المقاطعة RST5.5 و RST6.5 و RST7.5 وثلاثة خطوط تليفونات تخرج من سنترال فى شركة معينة كالتالى: تخيل أن واحدا من هذه الخطوط هو تليفون رئيس الشركة وهو الخط RST7.5 والخط الثانى هو خط نائب الرئيس وهو الخط RST6.5 وأما الخط الثالث فهو خط مدير شئون الأفراد وهو الخط RST5.5 . هناك مكتب استقبال يرد على التليفونات القادمة للأشخاص الثلاثة ويوجهها إلى الشخص المطلوب . فى وقت الدوام (العمل) يعمل سنترال الشركة ويكون نشطا وفى غير وقت الدوام لا يعمل هذا السنترال ، ذلك يكافىء تماما تنشيط هذه الخطوط بالأمر EI وحجبها أو إخمادها بالخط DI . كل واحد من الأشخاص الثلاثة (الرئيس ونائبه ومدير شئون الأفراد) يترك خبرا فى مكتب الاستقبال عما إذا كان يريد استقبال مكالمات أم لا ، ذلك يكافىء وضع الشفرة المناسبة فى البتات 0 إلى 2 فى مسجل التراكم ثم تنفيذ الأمر SIM . إفترض أن السنترال يعمل والاستقبال موجود أيضا فإنه سيرد علىالمكالمات ويوصلها أو يمنعها عن الأشخاص الثلاثة وذلك على حسب الأوامر التى تركوها عنده ، إن ذلك يكافىء تماما كون البت 3 تساوى واحدا فى الشفرة الموجودة فى المسجل A . أحيانا يكون السنترال يعمل ولكن الاستقبال مشغول ، فى هذه الحالة كل المكالمات القادمة ستوصل أو لا توصل على حسب آخر قائمة أوامر من الأشخاص الثلاثة دون أى اعتبار لأى تغيير يريده أى واحد منهم وهذا يكافىء حالة المسجل A قبل آخر مرة ينفذ فيها الأمر SIM وتكون البت 3 تساوى صفرا فى هذه الحالة . هناك ميزة فريدة للرئيس وهى أن عنده مفتاحا خاصا به بحيث عندما يكون هذا المفتاح ON ترد ماكينة إجابة ذاتية تقول "المدير مشغول الآن من فضلك اتصل مرة أخرى" . هذا يكافىء تماما البت 4 التى عندما تكون صفرا فإن الخط RST7.5 يكون فعالا وإذا كانت واحدا فإن هذا الخط يكون مقنعا أو محجوبا .
هناك الأمر RIM الذى يتكون من بايت واحدة والذى يقوم تقريبا بالعملية العكسية للأمر SIM حيث يقوم بتحميل المسجل A بثمانية بتات توضح حالة أقنعة المقاطعة . لذلك فإن هذا الأمر معناه "اقرأ أقنعة المقاطعة" Read Interrupt Masks . فى حالة قراءة أقنعة المقاطعة بالأمر RIM فإن بتات المسجل A تترجم محتوياتها كما فى شكل (13-5) حيث الثلاثة بت الأولى 0 إلى 2 هى حالة أقنعة خطوط المقاطعة RST5.5 و RST6.5 و RST7.5 كما تم تسجيلها باستخدام الأمر SIM مسبقا . البت الثالثة تبين حالة قلاب أو علم المقاطعة الذى رأينا كيف نتحكم فيه بالأمرين EI و DI . افترض أن المعالج يقوم الآن بخدمة مقاطعة للخط RST7.5 ، وفى أثناء ذلك افترض أن الخط RST6.5 طلب المقاطعة ، ماذا سيفعل المعالج ؟ إن المعالج يضع الخط RST6.5 فى حالة إنتظار

Pending إلى حين الانتهاء من خدمة المقاطعة RST7.5 . فى هذه الحالة يستطيع المبرمج كتابة بعض الأوامر فى آخر برنامج خدمة مقاطعة الطرف RST7.5 يجعل المعالج يذهب إلى خدمة المقاطعة RST6.5 بدلا من العودة للبرنامج الأساسى . البت رقم 5 فى مسجل التراكم تعكس هذه الحالة ، فإذا كان الطرف RST6.5 فى حالة انتظار فإن هذه البت تكون واحدا وتكون صفرا فى غير ذلك . البت رقم 4 تمثل الانتظار لطرف المقاطعة RST5.5 والبت رقم 6 تمثل الانتظار لطرف المقاطعة RST7.5 . البت الأخيرة فى مسجل التراكم كما فى شكل (13-5) تمثل البيانات المدخلة تتابعيا إن وجدت حيث سنترك الحديث عن الإدخال والإخراج التتابعى للبيانات الآن .



شكل (13-5) معنى بتات المسجل A فى حالة تنفيذ الأمر RIM

شكل (6-13) يبين مثالا على إختبار المقاطعة RST6.5 فى نهاية برنامج مقاطعة الخط RST7.5 بحيث إذا كانت فى حالة انتظار يذهب المعالج لخدمتها قبل الرجوع للبرنامج الأساسى .

```
                        ; RST7.5 نهاية برنامج خدمة مقاطعة الخط
RIM                     ;قراءة أقنعة المقاطعة
MOV B,A                 ; B تخزين معلومات الأقنعة فى المسجل
ANI 20H                 ; RST6.5 اختبار لمعرفة إذا كان الخط
                        ; فى الإنتظار , أى البت خمسة من A تساوى واحد
JNZ NEXT
EI                      ; RST6.5 لا ينتظر , نشط المقاطعة
RET                     ; عودة للبرنامج الأساسى
 NEXT: MOV  A,B         ; A أعد حالة الأقنعة للمسجل
       ANI 0DH          ; RST6.5 تنشيط المقاطعة قد تكون مقنعة
       ORI 08H          ; تنشيط البتات 0إلى 2 قد تكون خاملة
       SIM              ; سجل الصورة الجديدة للأقنعة
       JMP SERV6.5      ; RST6.5 إقفز إلى برنامج خدمة المقاطعة
```

شكل (6-13) خدمة RST6.5 من برنامج خدمة RST7.5

مثال 13-3

افترض أنه بعد تنفيذ الأمر RIM وجدت المحتويات التالية فى المسجل A :

D7 D6 D5 D4 D3 D2 D1 D0

0 1 0 0 1 0 0 1 = 49H

إن ذلك يعنى أن المقاطعة RST5.5 هى المحجوبة (D0=1) وأما المقاطعة RST6.5 و RST7.5 فنشيطان (D2D1=00) ، كذلك فإن علم المقاطعة IF نشيط أى أن الأمر EI مازال سارى المفعول ، والمقاطعة RST7.5 فقط هى التى فى حالة انتظار (D6D5D4=100) كما أن هناك 0 قادم من الطرف SID (D7=0) .

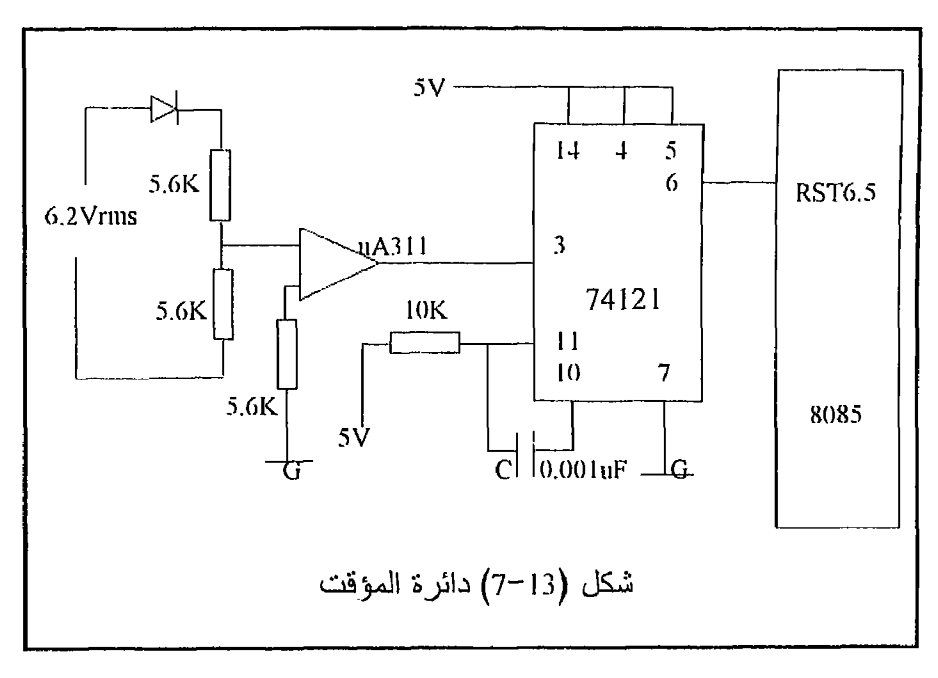
مثال 13-4

المطلوب عمل مؤقت timer لزمن مقداره دقيقة واحدة باستخدام التردد 60 هرتز الناتج من خط القدرة . هذا المؤقت يعرض الزمن على 3 خانات واحدة للدقائق وإثنتان للثوانى بحيث تعرض خانة الدقائق واحدا (أى دقيقة واحدة) ثم يبدأ العد

التصاعدى للثوانى إلى أن تصل إلى 60 فترجع إلى الصفر مرة أخرى وتستمر خانة الدقائق تعرض واحدا وتبدأ الثوانى فى العد التصاعدى ، وهكذا .

شكل (13-7) يبين الدائرة المستخدمة لهذا الغرض وشكل (13-8) يبين البرنامج المقترح لذلك . تعتمد الدائرة المستخدمة على تحويل جهد القدرة من 110 فولت إلى 6 فولت باستخدام محول خافض للجهد ثم تقسيم هذا الجهد وإدخاله على مقارن mA311 ليحول الموجة الجيبية إلى موجة مربعة TTL . تستخدم هذه الموجة لعمل إثارة trigger للشريحة 74121 التى تخرج نبضة TTL يمكن التحكم فى عرضها باستخدام المكثف C والمقاومة R . هذه النبضة الخارجة من الشريحة 74121 توصل على الطرف RST6.5 حيث ستعطى 60 نبضة مقاطعة فى الثانية الواحدة على هذا الطرف .

لكى نفهم البرنامج سنسير معه بالخطوات التالية :



شكل (13-7) دائرة المؤقت

أولا: عندما ستجىء نبضة مقاطعة على الطرف RST6.5 من الشريحة 74121 فإن المعالج سيقفز إلى العنوان 0034H كما أوضحنا فى شكل (13-2) . هذا العنوان يقع فى أول صفحة من الذاكرة حيث كل صفحة من الذاكرة تحتوى 256 بايت (FFH) وعادة ما يكون هذا الجزء من الذاكرة مشغولا ب EPROM تحتوى بعض البرامج الخاصة بتشغيل الميكروكومبيوتر . إذا كان الأمر كذلك فإن مبرمج ال EPROM عادة يسجل فى مثل هذه الأماكن أمر قفز إلى مكان آخر بحيث يكون هذا المكان فى ال RAM . لذلك يجب عليك أن تقرأ كتالوج الجهاز

الذى تتدرب عليه وتعرف منه أمر القفز الموجود عند العنوان 0034 أين يقفز فى ال RAM . سنفترض أن هذا العنوان هو أى عنوان وسنعطيه الرمز xxxx حيث سنكتب عند هذا العنوان برنامج خدمة المقاطعة .

```
0034 JMP xxxx        ;
;---------------Main program  هذا هو البرنامج الأساسى---------
LXI SP,STCK          ; STCK تحديد بداية المكدسة بالعنوان
MVI A,1DH            ; الرقم 1D ينشط المقاطعة RST6.5 ويخمد الباقى
SIM                  ; RST6.5 تنشيط المقاطعة
LXI B,0000H          ; تصفير المسجلين C, B للدقائق والثوانى
MVI D,3CH            ; الرقم 3CH يعادل 60 وهى عدد الثوانى
EI   ; تنشيط المقاطعة قبل الدخول فى برنامج إظهار الأرقام
DSPLY: MOV A,B
OUT 00               ; 00 إظهار الدقائق على البوابة
MOV A,C
OUT 01               ; 01 اظهار الثوانى على البوابة
JMP DSPLY
xxxx: DCR D          ; RAM بداية برنامج خدمة المقاطعة فى ال
EI                   ; تنشيط المقاطعة قبل الرجوع للبرنامج الأساسى
RNZ                  ; 60 عودة طالما أن عدد المقاطعات لم يصل إلى
DI                   ; أخماد المقاطعة إذا وصل العدد إلى 60 مقاطعة
MVI D,3CH            ; تحميل المسجل D بالرقم 60 ثانية مرة أخرى
INC C                ; زد الثوانى بمقدار واحد
MOV A,C
DAA                  ; وضع الثوانى فى الصورة العشرية
MOV C,A
CPI 60               ; هل وصل عدد الثوانى إلى 60 ثانية
EI                   ; تنشيط المقاطعة قبل العودة للبرنامج الأساسى
RNZ                  ; عودة طالما لم نصل إلى 60 ثانية
DI                   ; إخماد المقاطعة إذا وصل عدد الثوانى إلى 60 ثانية
MVI C,00             ; تصفير عداد الثوانى
MVI B,01             ; حمل المسجل B بدقيقة (زمن التأخير)
EI;                  ; تنشيط المقاطعة قبل العودة
RET  ;   عودة للبرنامج الأصلى
```

شكل (13-8) برنامج المؤقت باستخدام المقاطعة

ثانيا : أهم جزء فى البرنامج الأساسى هو حلقة لا نهائية تخرج على بوابتى الإخراج 00 و 01 قيمة الثوانى التى تتغير كل ثانية وقيمة الدقائق التى هى واحد باستمرار والتى يمكن التحكم فيها من البرنامج ، البوابات غير مبينة فى شكل (13-7) للتبسيط فقط .
ثالثا : مع كل نبضة مقاطعة على الطرف RST6.5 سيقفز المعالج من الحلقة اللانهائية إلى العنوان 0034 ومنه إلى العنوان xxxx حيث سيبدأ من هناك تنفيذ برنامج خدمة المقاطعة .
رابعا : فى برنامج خدمة المقاطعة سيكون هناك عداد تزداد محتوياته بمقدار واحد مع كل نبضة مقاطعة بحيث أنه طالما أن محتويات هذا العداد لم تصل إلى 60 فإن المعالج يرجع إلى البرنامج الأساسى فى الحلقة الانهائية حيث يستأنف عمله كما فى الخطوة (ثانيا) .
خامسا : مع تكرار المقاطعة سيصل العداد فى برنامج الخدمة إلى 60 حيث عندها يقوم برنامج الخدمة بتعديل قراءة الثوانى بزيادتها بمقدار ثانية واحدة .
سادسا : إذا وصل عدد الثوانى إلى 60 يكون قد مضى دقيقة ، عندها يرجع عداد الثوانى إلى الصفر لتبدأ العملية من جديد .
شكل (13-2) يبين كيفية استقبال المعالج لنبضات المقاطعة على الخطوط RST5.5, RST6.5, RST7.5 حيث نلاحظ من هذا الشكل أنه بالنسبة للخط RST7.5 بالذات يكفى أن تصعد النبضة من صفر low إلى واحد high لكى يستقبل المعالج هذه الإشارة وذلك لوجود ماسك فى مدخل هذا الخط ، الخطان RST5.5, RST6.5 لابد أن تبقيا واحدا high إلى أن يقبلها المعالج وإلا لو أنها نزلت إلى الصفر low قبل أن يقبلها المعالج فإن هذه المقاطعة لن تخدم بواسطة المعالج لعدم وجود ماسك فى مدخل كل خط من هذه الخطوط .

13-4-2 الخط TRAP

خط المقاطعة TRAP يتميز بميزة خاصة عن الخطوط السابقة وهى أن هذا الخط لا يمكن حجبه أو إخفاؤه أو تعطيله بأى واحد من الأوامر السابقه وهى SIM أو DI وكذلك لا يحتاج للأمر EI لتنشيطه . لذلك فإن هذا الخط يسمى Nonmakable interrupt أو المقاطعة التى لا يمكن حجبها . عادة يستخدم هذا الخط كما أشرنا سابقا فى عمليات المقاطعة الخطيرة مثل الحريق أو إنقطاع التيار الكهربى أو غير ذلك من العمليات الصناعية الخطيرة . هذا الخط كما هو موضح فى شكلى (13-2 و 13-3) لابد وأن يرتفع من صفر إلى واحد ويستمر واحدا high إلى أن يقبله المعالج وإذا نزل إلى الصفر قبل أن يقبله المعالج فلن يكون له أى تأثير .

13-4-3 الخط INTR

آخر خط من خطوط المقاطعة فى المعالج 8085 هو الخط INTR . هـذا الخـط يعتبر من خطوط المقاطعة التى يمكن إخفاؤها أو وضع قنـاع عليـها أى أنـها maskable عن طريق الأمر DI ولا تقبل أو يتم خدمتها إلا إذا كان علم المقاطعة IF فعالا عن طريق الأمر EI كما فى شكل (13-3) . سنحاول فهم ما يقوم بـه المعالج وما يحتاج إليه من دوائر خارجية من خلال تتبعنا للخطوات التى يقوم بها المعالج عندما يشعر بأن الخط INTR واحد high وهذه الخطوات كما يلى :

الخطوة الأولى : فى أثناء تنفيذ كل أمر من أوامر أى برنـامج يقـوم المعـالج باختبار الخط INTR هل يساوى واحدا أم صفرا ، وذلك من تلقاء نفسـه وتبعـا لتركيبه المنطقى .

الخطوة الثانية : إذا كان هذا الخط صفرا فإن المعالج يذهب لتنفيذ الأمر التـالى وأما إذا كان واحدا high فإن ذلك يعنى طلبا للمقاطعة . عندهـا إذا كـان علـم المقاطعة يساوى واحدا عن طريق الأمر EI فإن المعالج سيقبل المقاطعة ويكمـل تنفيذ الأمر الحالى ويدفع بمحتويات عداد البرنامج إلى المكدسة stack حتى يتمكن من العودة إلى نفس المكان الذى خرج منه .

الخطوة الثالثة : يقوم المعالج بجعل علم المقاطعة IF صفرا لمنع أى مقاطعة مـن المصادر الأخرى (بالطبع ما عدا المقاطعـة مـن الخـط TRAP) ثـم يجعـل الخط $\overline{INTA}$ فعالا بجعله يساوى صفرا low . الخط $\overline{INTA}$ عندما يكـون فعـالا يعنى أن المعالج قد قبل المقاطعة Interrupt Acknowledge وهـو فـى إنتظـار تحديد المكان الذى سيقفز إليه ليبدأ تنفيذ برنامج خدمة المقاطعة وذلك لأنه وكمـا هو مبين فى شكل (13-9) فإن هذا الخط ملحق به ثمانية عناوين يمكن القفز إلـى أى واحد منها يتم تحديده للمعالج عند طلب المقاطعة . هذه العناوين مرقمة مـن صفر إلى سبعة ويتم القفز إلى أى واحد منها عن طريق تنفيذ الأمر RST n حيث n هى رقم العنوان . هذا الأمر يعطى للمعالج عن طريق دائرة خارجية سـنعرفها بعد قليل حيث تستخدم الإشارة low على الخط $\overline{INTA}$ لتنشيط هذه الدائرة لتعطى المعالج الأمر RST n ورقم العنوان المطلوب القفز إليه .

الخطوة الرابعة : عندما يقرأ المعالج الأمر RST n من على مسار العناوين فإنـه يقفز إلى العنوان المحدد بهذا الأمر ليبدأ تنفيذ برنامج خدمة المقاطعة من هناك .

الخطوة الخامسة : قبل الانتهاء من برنامج خدمة المقاطعة يجب أن نضع الأمر EI ثم الأمر RET فى نهايته حتى ننشط المقاطعة فيصبح المعالج جاهزا لاستقبال المقاطعات الأخرى عند عودته إلى مكانه الذى خرج منه فى البرنامج الأصلـى عن طريق إسترداد محتويات عداد البرنامج التى دفعها إلى المكدسة .

13-4-4 كيف يتم تحديد العنوان الذى سيتم القفز إليه فــــى حالــة المقاطعة INTR ؟

كما ذكرنا فى الخطوة الرابعة فإن المعالج بعد أن يجعل الخط $\overline{INTA}$ فعالا يكون فى إنتظار قراءة أمر معين من على مسار البيانات وهذا الأمر هو الأمر RST n الذى يتكون من بايت واحدة تحتوى شفره لرقم العنوان الذى سيتم القفــــز إليـــه . شكل (13-9) يبين جدولا لجميع حالات الأمر RST n والعنوان المصاحب لكـــل حالة حيث نلاحظ من هذا الجدول أن البتات D3 و D4 و D5 تمثل الرقم n فـــى كل حالة . شكل (13-10) يبين دائرة مقترحة لإدخال شفرة الأمر RST n علـــى مسار البيانات فور نزول الخط $\overline{INTA}$ إلى الصفر . تتكون هذه الدائرة أساسا من أى شريحة بوابات ثلاثية المنطق ولتكن الشريحة 74244 مثلا والتى تحتوى علـى ثمانى بوابات من هذا النوع . يوصل خط تنشيط هذه البوابات بالخط $\overline{INTA}$ القادم من المعالج ويوصل خرج الشريحة على مسار البيانات . توصل جميع خطـــوط الدخل بالواحد high ما عدا الثلاثة خطوط D5, D4, D3 فتوصل علـــى مفـــاتيح تضبط بحيث تعطى الشفرة المناسبة للرقم n المطلوب مع الأمر RST n كما فـــى الجدول المبين فى شكل (13-9) . شكل (13-10) يبين هذه الدائرة وقد تم ضبــط هذه المفاتيح الثلاثة لتعطى الأمر RST 5 .

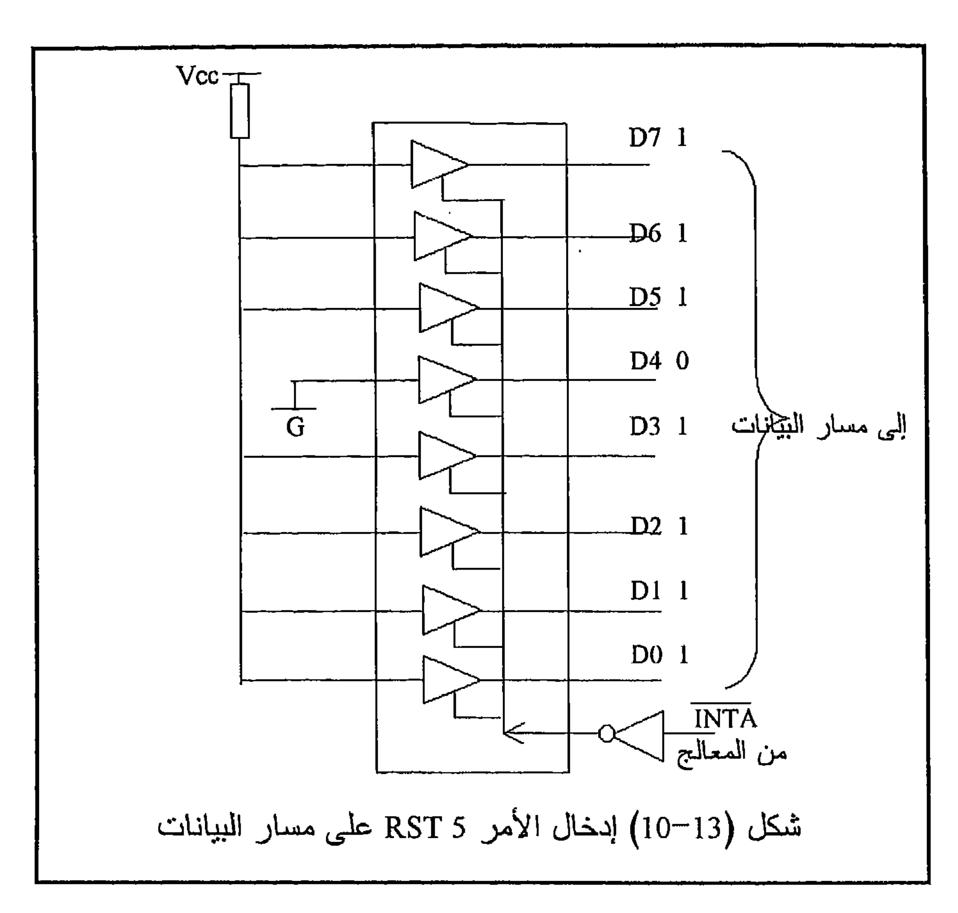
الأمر RST n	D7D6	D5D4D3	D2D1D0	شفرة الأمر	العنوان
RST 0	1 1	0 0 0	1 1 1	C7	0000
RST 1	1 1	0 0 1	1 1 1	CF	0008
RST 2	1 1	0 1 0	1 1 1	D7	0010
RST 3	1 1	0 1 1	1 1 1	DF	0018
RST 4	1 1	1 0 0	1 1 1	E7	0020
RST 5	1 1	1 0 1	1 1 1	EF	0028
RST 6	1 1	1 1 0	1 1 1	F7	0030
RST 7	1 1	1 1 1	1 1 1	FF	0038

شكل (13-9) الأمر RST n شفرته وعناوين القفز

لاحظ أن الدائرة الموجودة فى شكل (13-10) تكافىء تماما بوابـــة إدخـــال تــم الاستغناء عن عمليات تشفير عنوان لها واستخدم الخط $\overline{INTA}$ كخط فعالية لــــهذه البوابة بحيث أن هذه البوابة ستضع دخلها على مسار البيانات فقط عندما يكـــون الخط $\overline{INTA}$ فعالا .

إن الأمر RST n يمكن تنفيذه من خلال البرنامج وليس بالضرورة أن ينفذ عـــن طريق إدخاله على مسار البيانات كما رأينا ، فإنه يمكن فى أى مكان فى البرنامج

أن تنفذ الأمر RST n حيث سيقفز المعالج إلى العنوان المتوافـق مـع الرقـم n وسينفذ برنامج المقاطعه الموجود هناك تماما كما لو كان المعالج تمت مقاطعتـه من الخارج من على الخط INTR وهذه تسمى مقاطعة عـن طريـق البرمجـة software interrupt . بذلك نكون قد إنتهينا من الطرق المختلفة لمقاطعة المعـالج 8085 .



شكل (13-10) إدخال الأمر RST 5 على مسار البيانات

13-5 مقاطعة المعالج Z80

13-5-1 الخط $\overline{NMI}$

هناك خطان أساسيان لمقاطعة المعالج Z80 وهما الخط $\overline{NMI}$ على الطرف رقـم 17 فى الشريحة والخط $\overline{INT}$ على الطرف رقم 16. بالنسبة للخط $\overline{NMI}$ والـذى يعنى مقاطعة غير محجوبة Nonmaskable Interrupt فإنه فعال عندمـا يكـون صفرا low ولا يهم أن يبقى صفرا لكى يجيب المعالج على المقاطعة ولكن يكفـى

أن يهبط الجهد على هذا الطرف من الواحد إلى الصفر لكى يسجل المعالج طلب المقاطعة ، أى أن الحافة الهابطة هى الحافة المهمة لهذا الطرف . عندما ينتهى المعالج من تنفيذ أى أمر فإنه يختبر خط المقاطعة $\overline{NMI}$ فإذا كان فعالا يقوم فورا بعمل الآتى :

1. يقوم المعالج بدفع محتويات عداد البرنامج فى المكدسة حتى يتمكن من العودة إلى نفس المكان الذى خرج منه فى البرنامج الأساسى قبل طلب المقاطعة مثله فى ذلك مثل البرامج الفرعية subroutines.
2. القلاب IFF1 يعتبر علم المقاطعة فى المعالج Z80 بحيث لا يسمح بالمقاطعة إلا إذا كان هذا القلاب يساوى واحد . فى البرنامج الأساسى عادة يقوم المبرمج بوضع هذا القلاب إما واحد أو صفرا على حسب ظروف البرنامج إذا كان سيسمح بالمقاطعة من على الخط $\overline{INT}$ أم لا ، ولذلك فإن المعالج يقوم بتخزين قيمة هذا القلاب فى قلاب آخر IFF2 حتى إنه عندما يرجع إلى البرنامج الأساسى بعد خدمة المقاطعة يسترجع القيمة الأصلية للقلاب IFF1 من القلاب IFF2 والتى كانت قائمة قبل المقاطعة . يقوم المعالج بهذه العملية نتيجة لأنه سيغير من قيمة القلاب IFF1 كما سنرى .
3. يوضع القلاب IFF1 يساوى صفرا وبذلك تمنع أو تحجب أى مقاطعة من على الطرف $\overline{INT}$ وذلك بالطبع نتيجة لأهمية المقاطعة $\overline{NMI}$ وأولويته على المقاطعة $\overline{INT}$ ولذلك فإنه لا يسمح بالمقاطعة من على هذا الطرف إلا فى حالات الطوارىء كما ذكرنا سابقا ولذلك فإنه بمجرد أن يبدأ المعالج فى خدمة هذه المقاطعة فإنه تمنع أى مقاطعة أخرى من أى طرف آخر .
4. يقفز المعالج إلى العنوان 0066H فى الذاكرة والذى من المفروض أن يبدأ من عنده برنامج خدمة المقاطعة للطرف $\overline{NMI}$.
5. لابد وأن ينتهى برنامج خدمة المقاطعة بالأمر RTI والذى يعنى عودة من مقاطعة Return from Interrupt والذى على أثره يقوم المعالج باسترجاع محتويات عداد البرنامج من المكدسة فيرجع إلى البرنامج الأساسى ولنفس المكان الذى تمت عنده المقاطعة ، كما يقوم باسترجاع محتويات القلاب IFF1 من القلاب IFF2 بذلك يعود كل شىء إلى حالته قبل المقاطعة تقريبا .
6. إذا كان برنامج خدمة المقاطعة سيغير من قيمة أى من المسجلات المهمة فى البرنامج الأساسى فإن مسئولية دفع قيم هذه المسجلات إلى المكدسة فى بداية برنامج خدمة المقاطعة ثم استرجاعها مرة أخرى فى نهايته تقع على المبرمج وذلك لأن المعالج لا يقوم بتخزين كافة المسجلات .

7 – إذا بقى الخط $\overline{NMI}$ فعالا حتى عند الانتهاء من الخدمة والعودة إلى البرنامج الأساسى فإن ذلك سيسبب مقاطعة أخرى وهذه بالطبع حالة غير مرغوب فيها ، لذلك فإن إزالة طلب المقاطعة من على الخط $\overline{NMI}$ بعد الانتهاء من برنامج

الخدمة تقع أيضا على عاتق المستخدم وعليه أن يأخذها فـــى اعتبـــاره . أيســـر الطرق لذلك هى استخدام نبضة – واحد صفر واحد – كالمبينة فى شـــكل (13-11) لطلب المقاطعة وذلك لأنه كما ذكرنا فإن الخط حساس للحافة الهابطــــة ولا فائدة من ابقاء الخط أو مسكه على حالة الصفر ، لذلك فإن استخدام مثـــل هـــذه النبضة ستغنى المستخدم عن إضافة دوائر أخرى تزيل طلب المقاطعة إذا كـــان سيبقى ممسوكا على الصفر .

1

0

الحافة الفعالة

شكل (13-11) استخدام مثل هذه النبضة لا يحتاج لإزالة طلب المقاطعة بعد خدمتها

13-5-2 الخط $\overline{\text{INT}}$

خط المقاطعة $\overline{\text{INT}}$ يدخل على الطرف رقم 16 فى المعالج Z80 . المقاطعة علـــى هذا الخط يمكن حجبها باستخدام الأمر DI أى Disable Interrupt والذى يجعـــل علم المقاطعة IFF1 يساوى صفرا ، بذلك لا يمكن قبول أى مقاطعة علـــى هـــذا الخط . يمكن تنشيط المقاطعة مرة ثانية باستخدام الأمر EI والذى يعنـــى Enable Interrupt حيث يجعل علم المقاطعة IFF1 يساوى واحدا وبذلك فإن أى مقاطعـــة على الخط $\overline{\text{INT}}$ تقبل ويقوم المعالج بخدمتها . الخط $\overline{\text{INT}}$ يكون فعالا عندما يكون صفرا low ولابد لكى تقبل المقاطعة من على هذا الخط أن يظل صفرا إلـــى أن تقبل المقاطعة ، أى أن هذا الخط ينشط أو يكون فعالا بالمستوى صفـــر وليـــس بالحافة كما رأينا فى حالة المقاطعة $\overline{\text{NMI}}$. هناك ثلاثة طرق للتعامل مع الخـــط $\overline{\text{INT}}$ وهى الطريقة 0 والطريقة 1 والطريقة 2 ويمكن تحديد أو إختيـــار أى مـــن الطرق الثلاثة باستخدام واحد من الأوامر التالية ، IM2, IM1, IM0 وذلـــك مـــن داخل البرنامج الأساسى . لاحظ أن المعالج Z80 عند بدايـــة تشـــغيله أو إعـــادة وضعة أى RESET يكون فى الطريقة 0 .

الطريقة 0 للتعامل مع الخط $\overline{\text{INT}}$: هذه الطريقة تماثل تماما طريقـــة مقاطعـــة المعالج 8085 من على الخط INTR والتى سبق شرحها بالتفصيل حيث أنه عندما يستقبل المعالج Z80 طلب مقاطعة على الخط $\overline{\text{INT}}$ (طرف 16) فإنه يقفـــز إلـــى واحد من ثمانية عناوين فى الذاكرة يقوم المستخدم بتحديده للمعالج عنـــد طلـــب

المقاطعة . هذه العناوين الثمانية سبق تحديدها فى شكل (13–9) ويمكن مراجعتها للتذكرة (لذلك ننصح بقراءة الجزء 13–4–3 والجزء 13–4–4) . عندما يستقبل المعالج Z80 طلب مقاطعة على الخط $\overline{INT}$ وعندما يكون فى الطريقة 0 أى أنه سبق تنفيذ الأمر IM0 فإن المعالج يقوم بتنفيذ الخطوات التالية:

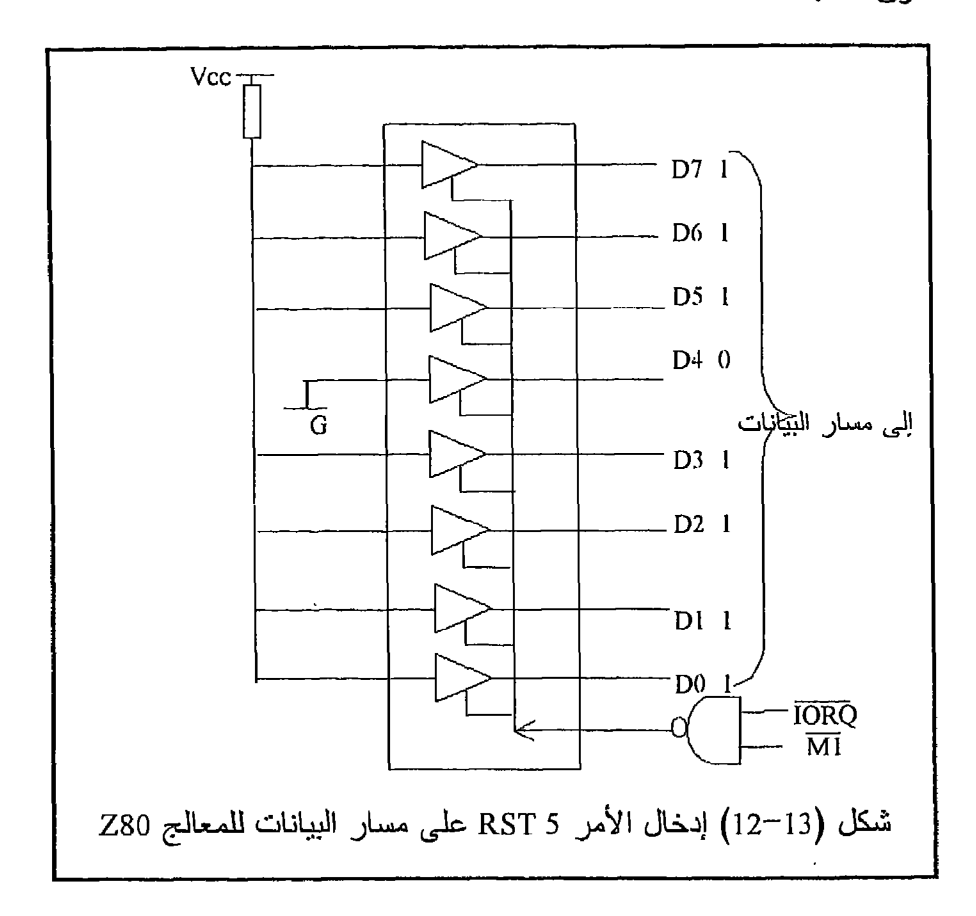
1. يقوم المعالج بتصفير علم المقاطعة IFF1 لحجب أى مقاطعات أخرى من أى أجهزة أخرى تطلب على نفس الخط $\overline{INT}$.
2. يقوم المعالج بجعل الخطين $\overline{IORQ}$ و $\overline{M1}$ فى حالة فعالية ، أى أن كل واحد من هذين الخطين يصبح صفرا low ، والحالة الوحيدة التى يصبح فيها هذان الخطان صفرا معا وفى نفس الوقت هى حالة المقاطعة على الخط $\overline{INT}$ بالطريقة 0 والتى نحن بصددها الآن . لاحظ أن الخط $\overline{IORQ}$ معناه طلب قراءة من جهاز إدخال ويقوم المعالج بتنشيط هذا الخط لأنه سيكون فى هذه الحالة فى إنتظار قراءة الأمر RST n كما شرحنا فى حالة المعالج 8085 . وأما الخط $\overline{M1}$ فإنه يكون فعالا فقط عند قراءة شفرة أى أمر ، وحيث أن المعالج فى حالتنا هذه بالذات يقرأ شفرة الأمر RST n من على بوابة إدخال فإنه سيكون هو أيضا فعالا فى نفس اللحظة التى سيكون فيها الخط $\overline{IORQ}$ فعالا .
3. على المستخدم أن يستغل ظاهرة أن الخطين $\overline{M1}$ و $\overline{IORQ}$ يكونان صفرا معا فى هذه الحالة فقط لتشغيل بوابة إدخال يدخل عليها شفرة الأمر RST n والذى يحدد للمعالج أى واحد من عناوين القفز الثمانية سيقفز إليه . هذه البوابة موضحة فى شكل (13–12) . لاحظ أن الفرق الوحيد بين المعالج Z80 والمعالج 8085 هو فقط فى طريقة تشغيل هذه البوابة حيث كما رأينا فى حالة المعالج 8085 يستخدم الخط $\overline{INTA}$ لتشغيل هذه البوابة كما فى شكل (13–10) .
4. يقرأ المعالج بوابة الإدخال ويفك شفرة الأمر RST n ليعرف العنوان الذى سيتم القفز إليه .
5. يقوم المعالج بدفع محتويات عداد البرنامج فى المكدسة .
6. يقفز المعالج إلى برنامج خدمة المقاطعة بناء على العنوان الذى قرأه فى الخطوة 4 .
7. بعد الانتهاء من تنفيذ برنامج خدمة المقاطعة يرجع المعالج إلى نفس المكان فى البرنامج الأصلى عن طريق جلب محتويات عداد البرنامج مرة ثانية من المكدسة .

الطريقة 1 للتعامل مع الخط $\overline{INT}$: هذه الطريقة بسيطة جدا حيث أنها تشبه تماما طريقة المقاطعة من على الخط $\overline{NMI}$ حيث أن هناك عنوانا واحدا فقط وهو العنوان 0038H يتم القفز إليه فى حالة المقاطعة بهذه الطريقة . لاحظ أنه لابد من تنفيذ الأمر IM1 قبل طلب المقاطعة فى البرنامج الأساسى ، كما أن علم المقاطعة لابد وأن يكون واحد ، أى أنه تم تنفيذ الأمر EI أيضا لتنشيط هذا العلم وحتى

يمكن قبول المقاطعة . عند المقاطعة بهذه الطريقة أيضـا يقـوم المعـالج بدفـع محتويات عداد البرنامج إلى المكدسة وعلى المبرمج دفع باقى المسجلات إن أراد فى بداية برنامج خدمة المقاطعة حيث أن المعالج لا يقوم بذلك .

الطريقة 2 للتعامل مع الخط $\overline{INT}$: يدخل المعالج فى هذه الطريقة بنـاء علـى تنفيذه للأمر IM2 ، وعند إستقباله لطلب مقاطعة يقوم بالخطوات التالية :

1. يقوم المعالج بوضع صفر فى علم المقاطعة IFF1 لمنع أى مقاطعة أخرى مثل الطرق السابقة .



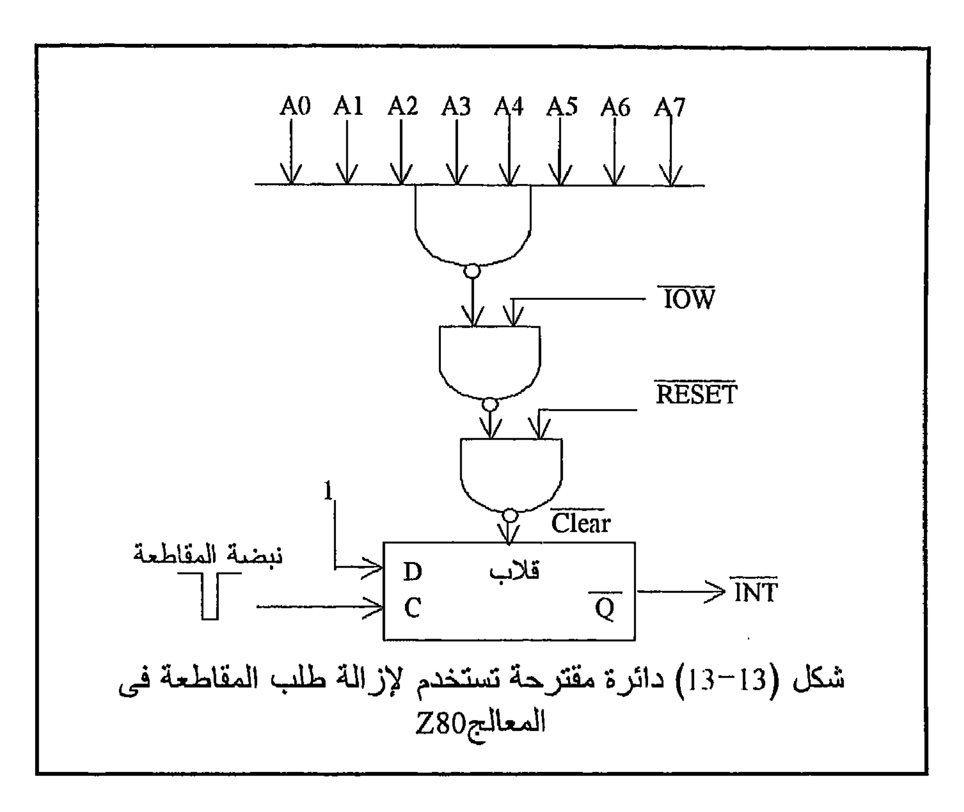
شكل (13-12) إدخال الأمر RST 5 على مسار البيانات للمعالج Z80

2. مثل الطريقة 0 يقوم المعالج بجعل الخطين $\overline{M1}$ و $\overline{IORQ}$ فعالين بجعل كـل منهما يساوى صفرا . على المبرمج مثل ما شرحنا فـى حالـة الطريقـة 0 أن يستغل ذلك لإدخال شفرة معينة إلى مسار البيانات مـن علـى بوابـة إدخـال كالموضحة فى شكل (13-12) .
3. يقوم المعالج بتكوين عنوان من 16 بتا وهذا العنوان يتكـون مـن جزأيـن : الجزء الأول من العنوان وهو A0 إلى A7 هو الشفرة التى تمت قراءتها من على

بوابة الإدخال كما فى الخطوة 2 وأما الجزء الثانى من العنوان A8 إلى A15 فهو محتويات المسجل I الموجود فى المعالج لهذا الغرض والذى يتكون من 8 بتـلت. عنوان برنامج خدمة المقاطعة فى هذه الحالة يكون موجودا فى البايت المحـــددة بالعنوان السابق والبايت التى تليها .

4. بعد معرفة هذا العنوان يقفز إليه المعالج لينفذ برنامج خدمة المقاطعة الموجود هناك وذلك بالطبع بعد دفع محتويات عداد البرنامج إلى المكدسة حتى يســـتطيع العودة إلى نفس المكان فى البرنامج الأصلى بعد الانتهاء مـــن برنـــامج خدمـــة المقاطعة .

5. يمكن أن نوضح ذلك بالمثال التالى : افترض أن محتويات المســـجل I هـــى 10H ، وعند قراءة بوابة الإدخال المقصودة وجد بها الرقم 45H . لذلك ســيكون المعالج العنوان التالى 1045H ويذهب إلى هذه البايت ليقرأ محتوياتها ولتكن مثلا 00H ثم يذهب إلى البايت التالية لها أى البايت 1046H فيقرأ محتوياتها ولتـــكن مثلا E1H . من محتويات هاتين البايتين يتكون لدى المعالج عنوان برنامج خدمة المقاطعة الذى يكون E100H فى هذا المثال .



شكل (13-13) دائرة مقترحة تستخدم لإزالة طلب المقاطعة فى المعالجZ80

كما رأينا فإنه لابد وأن يبقى الخط $\overline{INT}$ ممسوكا على الصفر حتى تتم خدمته بواسطة المعالج وإلا فإنه إن رجع إلى الواحد قبل أن يتعرف المعالج عليه فلن تتم خدمته وذلك لأن الخط $\overline{INT}$ من النوع الذى يكون فعالا على المستوى صفر وليس على أى حافة . لذلك فإنه يستحسن إدخال طلب المقاطعة على الطرف $\overline{INT}$ عن طريق قلاب كما هو مبين فى شكل (13-13) . وجود مثل هذا القلاب فى مدخل المقاطعة $\overline{INT}$ ومسكه على الصفر عند طلب المقاطعة يستلزم أن يقوم المستخدم بإزالة هذا الصفر (طلب المقاطعة) عند نهاية برنامج الخدمة . شكل (13-13) يبين أيضا دائرة مقترحة لهذا الشأن حيث يتم إخراج أى معلومة على بوابة الإخراج FFH مثلا التى نتيجة لتشفيرها تعطى واحد على الخط $\overline{INT}$ أى تزيل المقاطعة . أى أنه بمجرد تنفيذ الأمر OUT FFH فى نهاية برنامج خدمة المقاطعة سيزال طلب المقاطعة من على الطرف $\overline{INT}$.

13-6 تمارين

1. ما هو المقصود بطرق الأبواب لخدمة الأجهزة ؟
2. ما هو الفرق بين طريقة طرق الأبواب لخدمة الأجهزة والمقاطعة ؟
3. اشرح بعض التطبيقات التى تستخدم فيها المقاطعة ؟
4. ماذا يفعل أى معالج عادة حال إستقباله لأمر مقاطعة ؟
5. كم عدد خطوط المقاطعة لدى المعالج 8085 التى يمكن مقاطعته من عليها ؟
6. كم عدد خطوط المقاطعة لدى المعالج Z80 التى يمكن مقاطعته من عليها ؟
7. الخط TRAP فى المعالج 8085 ، ماذا يكافىء فى المعالج Z80 ؟
8. الأمران DI و EI للمعالج 8085 يستخدمان عادة فى برامج خدمة المقاطعة ، ما هو الغرض من استخدامهما ؟
9. اشرح دور الأمرين SIM و RIM فى المقاطعة على الخطوط RST7.5, RST6.5, RST5.5, ؟
10. ما هو الفرق بين المقاطعة على الخط INTR والخطوط الأخرى فى المعالج 8085 والمعالج Z80 ؟
11. اشرح كيفية تحديد مكان برنامج خدمة المقاطعة للمعالج 8085 فى حالة المقاطعة على الخط INTR ؟
12. اشرح كيفية إدخال شفرة الأمر RST n إلى المعالج 8085 و Z80 ؟
13. من أوامر الشريحة Z80 الأوامر ، IM0 و IM1 و IM2 , ماذا تعنى هذه الأوامر وهل لها نظير فى المعالج 8085 ؟ قارن بين الطرق الثلاثة لمقاطعة المعالج Z80 على الخط $\overline{INT}$ وما يناظرها فى المعالج 8085 ؟
14. قارن بين الشكلين (13-10) و (13-12) ؟

15. مطلوب إستخدام المعالج للتحكم فى نظام درجة حـــرارة غرفـــة فـــى أحـــد المصانع وإظهار هذه الدرجة على شريحتين ذات 7 قطع , 7 segment إرســـم هذا النظام وإكتب برنامجه مرة مستخدما نظام طرق الأبواب ومـــرة باســـتخدام المقاطعة ؟

الفصل الرابع عشر

التركيب الهيكلى للمعالج

Intel 8086/8088

14-1 مقدمة

نقدم فى هذا الفصل شرحا تفصيليا للتركيب الهيكلي للمعالج intel8086 وزميلــه المعالج intel8088 وذلك من الداخل حيث سندرس محتوياتهما مــن المســجلات والعدادات ومن الخارج حيث سنلقى نظرة سريعة على وظيفة كل طــرف مــن أطرافهما . إن هذين المعالجين يعتبران أهم المعالجات ذات ال 16 بــت وذلــك لاستخدامهما فى الحاسب IBM الذى كان من أول الحاسـبات الشـخصية التــى ظهرت فى السوق ثم سادت وفرضت نفسها على كل المســتخدمين للحاســبات . من ضمن هذا الجيل من المعالجات ظهر أيضا المعــالج MC68000 والمعــالج Z8000 ولكن كما ذكرنا كان أكثرها شيوعا وأوفرها حظا فــى الاســتخدام هــو المعالج intel8086 والذى نحن بصدد دراسته هنا .

المعالجين 8086/8088 كل منهما كما رأينا يتعامل مع بيانات مقدارهـــا 16 بت ، ولكن هناك نقطة خلاف أساسية ووحيدة بين هذين المعالجين وهى كيفية التعــامل مع هذه البيانات . إن المعالج 8086 يتعامل مع البيانات فى داخله وفى خارجـــه على أساس أنها 16 بت ، أى أنه له مسار بيانات خارجى مقداره 16 بت يســتطيع من خلاله التعامل مع الأجهزة الخارجية مثل الشاشة والذاكرة ولوحـــة المفــاتيح على أساس 16 بت ، فيرسل مثلا معلومة من 16 بت إلى الشاشة فى مرة واحــدة ويستقبل معلومة من 16 بت من أى بوابة إدخال . هذا المعالج ، 8086 يتعــامل داخليا أيضا بنفس الطريقة حيث تنتقل البيانات داخليا بيــن جميــع المســجلات بعضها البعض أو مع وحدة الحساب والمنطق على مسار بيانات مقداره 16 بــت أيضا . أما المعالج 8088 فإنه تماما مثل نظيره 8086 فى التعــاملات الداخليــة حيث ينقل البيانات داخله على مسار بيانات مقداره 16 بت ، بينما يختلــف عــن نظيره 8086 فى التعاملات الخارجية حيث أن مسار البيانات الخارجى له يتكــون من 8 بت فقط ، لذلك فإنه يرسل أو يستقبل بيانات 8 بــت فقــط مــع الأجــهزة الخارجية ، وهذه كما ذكرنا هى نقطة الخلاف الوحيدة بين المعالجين كما ســنرى فيما بعد . ونعتقد أن الرقم 6 فى المعالج 8086 يذكرنا بأنه يتعــامل دائمــا مــن خلال مسار بيانات 16 بت ، بينما الرقم 8 الأخير فى المعالج 8088 فإنه يذكرنــا بأن التعامل يكون من خلال مسار بيانات 8 بت فقط مع الأجـــهزة الخارجيــة . ويخطر ببالنا سؤال مهم هنا وهو: ما هو الداعى للمعالج 8088 إذا كان نظـــيره 8086 قد قام بالمهمة بكفاءة أحسن وبالتأكيد أسهل حيث يتعامل خارجيا وداخليـــا من خلال مسار بيانات 16 بت ؟ والإجابة على ذلك هى أن المعالجــات 16 بــت ظهرت فى منتصف الثمانينات وحلت محل المعالجات 8 بت فى جميع الحاســبات وجميع التطبيقات وبعد أن كانت المعالجات 8 بــت قــد انتشــرت فــى الســوق واستخدمت فى كثير من أجهزة الحاسبات وفى الكثير من التطبيقات أيضـــا ممــا

تسبب فى أن كل هذه الأجهزة القديمة (8 بت) أصبحت عديمة الفائدة بعد أن فكر كل المستهلكين فى الأخذ بالمعالجات الجديدة (16 بت) . لذلك كان التفكير فى معالج وسيط يقلل من حجم الخسارة فى عملية الانتقال إلى المعالج 8086 ، فكان الحل هو معالج يتعامل داخليا على أساس 16 بت للاستفادة بمميزات ال16 بت ويتعامل خارجيا على أساس 8 بت لتسهيل عملية المواجهة مع الأجهزة الخارجية الموجودة أصلا فى الحاسبات والتطبيقات التى كانت تتعامل مع المعالجات 8 بت وبأقل تكلفة ممكنة ، فكان ذلك المعالج الوسيط هو المعالج 8088 ، ولقد نزلت فى هذا الوقت أيضا الكثير من البرامج التى تقوم بتحويل برمجيات المعالجات 8 بت إلى صورة تناسب المعالجات 16 بت كمرحلة انتقال وحتى لا يعاد صياغة هذه البرمجيات من جديد .

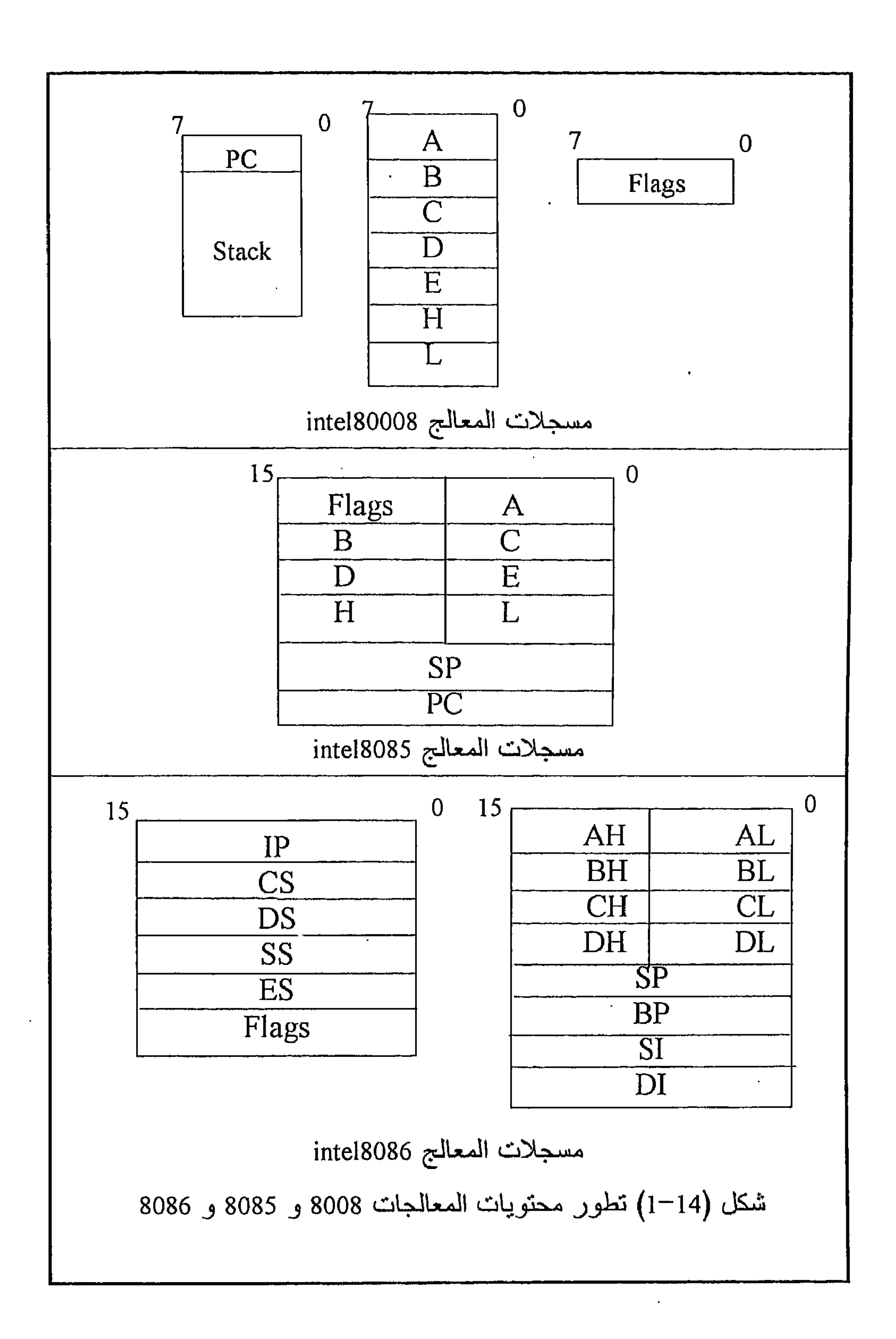
14-2 نظرة داخلية على محتويات المعالجين 8086/8088

كما علمنا من قبل وعند دراستنا لشرائح المعالجات 8 بت أن أى معالج فى النهاية يمكن النظر إليه على أنه مجموعة من المسجلات والعدادات بجانب وحدة الحساب والمنطق حيث أنها أهم المكونات الداخلية فى المعالج ، أما فكرة عمل المعالج ، أى معالج ، فهى (كما سبق وشرحناها أيضا) إحضار شفرات الأوامر من الذاكرة وتنفيذها بنفس التتابع المسجلة به فى البرنامج . أى أن الفكرة ثابتة ولكن التطور يكون دائما فى المكونات حيث تتغير المسجلات ووحدة الحساب والمنطق من 8 بت إلى 16 بت إلى 32 بت إلى 64 بت وتتغير تكنولوجيا التصنيع نفسها مع الزمن فتزداد السرعة بدرجة كبيرة ولكن فكرة العمل تظل كما هى ثابتة . شكل (14-1) يبين تطور المسجلات فى المعالجات 8008 و 8085 و 8086 . نلاحظ من هذا الشكل أن عدد المسجلات كان 6 مسجلات كل منها 8 بت بخلاف المركم فى المعالج 8008 وأما المعالج 8085 فيحتوى نفس العدد من المسجلات ولكن الجديد هو أن هذه المسجلات يمكن فى بعض العمليات ازدواجها واستخدامها كمسجلات 16 بت كما رأينا سابقا ولكن المركم كما هو 8 بت ، وأما فى المعالج 8086 فإنه يحتوى أيضا نفس العدد من المسجلات العامة والتى اختلفت أسماؤها قليلا لتناسب استخدامها كمسجلات 16 بت أو 8 بت ، فمثلا المسجل B أصبح اسمه BX فى حالة استخدامه كمسجل 16 بت أو BL فى حالة استخدام النصف الأول منه كمسجل 8 بت و BH فى حالة استخدام النصف الأعلى منه كمسجل 8 بت ، نفس الكلام مطبق على باقى المسجلات العامة وهى المسجلات C, D . الجديد أيضا أن المركم مطبق عليه نفس الكلام السابق فيمكن استخدامه كمسجل 16 بت (AX) أو مسجلين كل منهم 8 بت (AL, AH) . نلاحظ أيضا أن المكدسة stack كانت موجودة بداخل المعالج 8008 ثم انتقلت لتصبح

جزءا من الذاكرة يشار إلى قمتها أو أول مكان فاضى فيها بمحتويات المسجل SP أو مؤشر المكدسة وذلك فى المعالجات 8085 و 8086 . أما عداد البرنامج PC فكان 8 بت فى المعالج 8008 وأصبح 16 بتا فى المعالجات 8085 و 8086 وأصبح اسمه مؤشر الأوامر IP فى المعالج 8086 وهناك فرق كبير بين الاسم "عداد البرنامج" و"مؤشر الأوامر" بالرغم من التماثل فى الوظيفة ولكن فى حالة المعالج 8085 فإنه يتعامل مع ذاكرة مقدارها 64 كيلو بايت وأما فى حالة المعالج 8086 فإنه يتعامل مع ذاكرة مقدارها 1 ميجابايت من خلال فكرة زكية وهى فكرة تجزىء الذاكرة التى سنشرحها بعد قليل إن شاء الله . نلاحظ أيضا من شكل (14-1) أن المعالج 8086 يحتوى على مسجلات أخرى لم تكن موجودة فى سابقيه وهى المسجلات BP, SI, DI, CS, DS, SS, ES وكلها مسجلات 16 بت ستتعرف على وظيفة كل منها بعد قليل . نلاحظ أيضا أن مسجل الأعلام أصبح 16 بت أيضا بدلا من ثمانية مما ينبئ بأنه سيكون هناك الكثير من الأعلام وبالتالى مقدرة أكثر على البرمجة وعدد أكثر من الأوامر . من الملاحظات المهمة أيضا فى شكل (14-1) هى احتواء المعالج 8008 على المكدسة كمجموعة من المسجلات موجودة بداخل المعالج نفسه فى حين أصبحت هذه المكدسة جزء من الذاكرة فى المعالجات التى تلت ذلك مما أمكن معه تكبير المكدسة لأى كمية مطلوبة .

14-3 نظرة تفصيلية على مسجلات المعالج 8086/8088

لقد استخدم مصمموا المعالجين 8086/8088 فكرة زكية كان لها أكبر الأثر فى زيادة سرعة وكفاءة هذين المعالجين وهذه الفكرة هى انقسام هذه المعالجات إلى وحدتين أساسيتين لكل منهما وظيفة مختلفة تماما عن الوحدة الأخرى .
الوحدة الأولى هي وحدة التنفيذ Execution Unit (EU) وهى خاصة فقط بتنفيذ الأوامر ولا تتعداها لأى وظيفة أخرى ، والوحدة الثانية هى وحدة مواجهة المسارات Bus Interface Unit (BIU) وهذه أيضا لها وظيفة محددة وهى جلب الأوامر من الذاكرة ووضعها فى طابور أو قائمة انتظار queue فى انتظار التنفيذ عن طريق وحدة التنفيذ . هذا التقسيم فى الوظائف بين الوحدتين أتاح لوحدة التنفيذ أن تقوم فقط بتنفيذ الأوامر الموجودة فى قائمة الانتظار وفى أثناء انشغال وحدة التنفيذ بتنفيذ الأوامر تقوم وحدة مواجهة المسارات بجلب أوامر أخرى من الذاكرة ووضعها فى القائمة والعمل على أن تكون القائمة مملوءة دائما بالأوامر التى فى انتظار التنفيذ .



مسجلات المعالج intel80008

مسجلات المعالج intel8085

مسجلات المعالج intel8086

شكل (14-1) تطور محتويات المعالجات 8008 و 8085 و 8086

بذلك تم توفير وقت كبير كانت وحدة التنفيذ تتوقف فيه لحين الذهاب إلى الذاكـرة وإحضار الأوامر التى عليها الدور فى التنفيذ مما كان له أكبر الأثر فـى زيـادة سرعة هذه المعالجات بدرجة كبيرة . يمكن تمثيل مهمة كل من هاتين الوحدتيـن بمهمة المدير والسكرتارية حيث تكون السكرتارية هى المواجهة للعالم الخـارجى حيث تستقبل هى جميع الطلبات والمشاكل من الجمهور واعدادها فى ملـف كـل على حسب ترتيب قدومه ثم تعرض هذا الملف على المدير الذى يقوم بحل هـذه المشاكل فى حين تعمل السكرتارية على استقبال الطلبات الأخرى لحين أن ينتـهى المدير من تنفيذ ما معه من طلبات . بالطبع فإن ذلك يكون له أكبر الأثـر فـى سرعة تنفيذ الطلبات عن ما لو كان المدير وحده يقوم باستقبال كمية من الطلبـات ثم يجلس لتنفيذها وبعد أن ينتهى من تنفيذ هذه الكمية يقوم ليستقبل كمية أخـرى وهكذا .

بعض المسجلات داخل المعالج 8086 تتبع وحدة التنفيذ والبعض الآخر يتبع وحدة المواجهة على حسب وظيفة كل مسجل من هذه المسجلات حيث يجب أن نتوقـع أن جميع المسجلات العامة DX, CX, BX, AX ومسجل الأعلام والمسجلات SP DI, SI, BP, كلها تتبع وحدة التنفيذ وأما المسجلات ES, SS, DS, CS ومؤشـر الأوامر IP فتتبع وحدة المواجهة تبعا لوظيفة كل منها كما سنرى .

على ضوء ما ذكرنا فى المقدمة عن الفرق بين المعالجين 8086 و 8088 فإننـا يجب أن نتوقع أن وحدة التنفيذ EU فى كل من المعالجين ستكون نفسـها تمامـا وأما وحدة المواجهة BIU فستختلف فى المعالج 8086 عنها فـى المعـالج 8088 حيث أنها فى الأول ستتعامل مع مسار بيانات 16 بت بينما ستتعامل مـع مسـار بيانات 8 بتات فى المعالج الثانى وهذا هو وجه الاختلاف الأساسى بينهما .

14-3-1 المسجلات عامة الأغراض

يحتوى المعالج 8086 على أربع مسجلات عامة الأغراض كل منها 16 بتا وهـى المسجلات DX, CX, BX, AX . كل واحد من هذه المسجلات يمكـن التعـامل معها على أنها مسجلين كل منهم 8 بتات أو مسجل واحد 16 بـت . فـى حالـة التعامل معها على أنها مسجلات 8 بتات فـإن النصـف الأدنـى أو ذو القيمـة الصغرى Low significant half يرمز له دائما بالرموز التاليـة DL, CL, BL, AL أما النصف الأعلى أو ذو القيمة العليا High significant half فـيرمز لـه دائما بالرموز التالية DH, CH, BH, AH . لذلك عند وضع معلومة من 16 بـت مثل الرقم C351H فى المسجل BX مثلا فإن النصف الأعلى من المعلومة وهـو C3 يوضع فى النصف الأعلى من المسجل وهو BH وأما النصف الأدنـى مـن المعلومة وهو 51 فيوضع فى النصف الأدنى من المسجل وهو BL . فيما يلـى سنلقى نظرة سريعة على وظيفة كل مسجل من هذه المسجلات :

المسجل AX
المسجل AX هو المركم accumulator وكما سنرى عند دراستنا للغة الأسمبلى للمعالج 8086 فإن المركم لن تكون له نفس الأهمية التى رأيناها عند دراستنا للمعالجات 8 بت ، حيث هنا سنرى أنه يمكن إجراء أى عملية حسابية أو منطقية على أى مسجلين مع بعضهما البعض وليس من الضرورى أن يكون المركم واحد منهما ، كما أن نتيجة هذه العملية تكون دائما فى المسجل الأول فى الأمر ، فمثلا الأوامر التالية كلها صحيحة :

ADD AX, BX

حيث سيجمع محتويات المسجل AX مع المسجل BX ويضع النتيجة فى المسجل AX الذى هو المركم .

ADD CX, BX

حيث سيجمع محتويات المسجل CX مع المسجل BX ويضع النتيجة فى المسجل CX ، وهكذا . هذا ولا زالت عمليات الإدخال والإخراج باستخدام الأمرين IN و OUT تتم عن طريق المركم كما هو الحال فى المعالجات 8 بت ولكن بإمكانيات أكثر وكفاءة أحسن كما سنرى عند الدراسة التفصيلية لهذه الأوامر .

المسجل BX
إن المسجل BX بجانب كونه أحد المسجلات العامة التى تستخدم فى كل أغراض البرمجة مثل العمليات الحسابية والمنطقية وعمليات الإزاحة والدوران وغيرها فإن له وظيفة أخرى محددة وخاصة به عند تنفيذ بعض الأوامر مثل الأمر XLAT والذى ينشئ جدولا فى الذاكرة أول عنوان فيه هو الموجود فى المسجل BX وتتكون عناصر هذا الجدول بتخزين محتويات النصف الأدنى من المركم AL فى عناوين متتالية فى الذاكرة تتكون بجمع محتويات المسجل AL مع محتويات المسجل BX كما سنرى عند شرح أوامر لغة الأسمبلى فيما بعد . لذلك فإن المسجل BX يحتوى عنوان البداية أو القاعدة Base للجدول الذى يتكون بالأمر XLAT . كما أن المسجل BX يستخدم فى أغراض العنونة غير المباشرة حيث يمكن وضع العنوان المراد التعامل معه فى ذاكرة البيانات فيه .

المسجل CX
المسجل CX أيضا بجانب كونه أحد المسجلات العامة مثل المسجلين BX, AX فإن له أيضا مهمة محددة خاصة به عند تنفيذ بعض الأوامر . فمثلا عند تنفيذ الأمر LOOP والذى ينفذ حلقة أو مجموعة من الأوامر عدة مرات فإن عدد المرات المراد تنفيذها لهذه الحلقة يوضع فى المسجل CX . أى أنه عداد أو Counter للحلقات عند تنفيذ الأمر LOOP .

المسجل DX

هذا المسجل أيضا بجانب كونه أحد المسجلات العامة فله أيضا وظيفة محددة عند تنفيذ بعض الأوامر ، فعند تنفيذ أمر ضرب رقمين كل منهما 16 بت فإن النصف الأعلى most significant part من النتيجة يوضع في هذا المسجل (لاحظ أن النتيجة ستكون 32 بت) . كذلك عند تنفيذ بعض أوامر الإدخال والإخراج فإن المسجل DX يوضع به عنوان البوابة المراد الإخراج عليها أو الإدخال منها .
أى أن هذا المسجل DX أحد وظائفه الخاصة هى أنه يحتوى جزء من البيانات عند تنفيذ أوامر ضرب أو قسمة رقمين كل منهما 16 بتا . وعلى ذلك فإن الأربع مسجلات السابقة لها أسماء كما رأينا تطابق الرموز التى أطلقت عليها والوظيفة الخاصة المنوطة بكل واحد من هذه المسجلات والتى سنعيدها كما يلى:

Accumulator	AX	المركم
Base	BX	القاعدة
Counter	CX	العداد
Data	DX	البيانات

هناك أيضا مجموعة من المسجلات التى يمكن أن تدخل ضمن مجموعة المسجلات العامة حيث أنها تكون تحت تصرف المبرمج ولكنها لها وظيفة محددة أيضا فى عمليات البرمجة فهى تستخدم إما للإشارة إلى أماكن محددة فى الذاكرة Pointer أو تستخدم فى الفهرسة Index عند التعامل مع الذاكرة بهذه الطريقة . هذه المسجلات هى كالتالى :

مؤشر المكدسة أو المسجل Stack Pointer, SP

المكدسة stack هى جزء مقتطع من الذاكرة يخزن فيه عادة البيانات المهمة قبل القفز من البرنامج الأساسى إلى برنامج فرعى أو برنامج مقاطعة والتى ستكون هناك حاجة إليها عند العودة مرة ثانية إلى البرنامج الأساسى بعد إنهاء البرنامج الفرعى (انظر فصل البرامج الفرعية) أو الانتهاء من خدمة المقاطعة (انظر فصل المقاطعة) . من أهم هذه البيانات مثلا محتويات مؤشر الأوامر IP حتى يتسنى لنا العودة لنفس المكان الذى خرجنا منه فى البرنامج الأساسى وكذلك محتويات أى مسجل آخر قد نخاف من ضياعها أو تغيرها عند الخروج من البرنامج الأساسى مثل مسجل الأعلام والمركم . هذه البيانات تخزن فى المكدسة بالترتيب ويتم إستدعاؤها بنفس الترتيب على أساس أن آخر ما تم تخزينه يكون أول ما يتم استدعاؤه Last In First Out (LIFO) ولكى نعرف حدود هذه المكدسة فإن مسجل مؤشر المكدسة Stack Pointer (SP) يحتوى عنوان آخر مكان تم التخزين فيه فى هذه المكدسة .

مسجل مؤشر القاعدة Base Pointer, BP

أحد المسجلات العامة التى تستخدم لعنونة أو للإشارة على بداية مجموعة بيانات أو طابور array بيانات موجود فى المكدسة stack .

مسجلى الفهرسة SI, DI

يستخدمان فى عملية العنونة غير المباشرة (المفهرسة) فى الذاكرة indirect addressing كما سنرى عند دراسة طرق العنونة المختلفة .

14-3-2 المسجلات الخاصة

المسجلات الخاصة الموجودة فى المعالج 8086/8088 هى مسجل مؤشر الأوامر Instruction Pointer (IP) وأربع مسجلات خاصة بتجزىء الذاكرة سنطلق عليها اسم مسجلات التجزىء memory segmentation registers . عدد هذه المسجلات أربعة وهى : ES, SS, DS, CS . لكى نأخذ فكرة عن وظيفة هذه المسجلات ، لابد أن نعرف أولا ما هو المقصود بتجزىء الذاكرة ؟ ولماذا يتم تجزىء الذاكرة؟

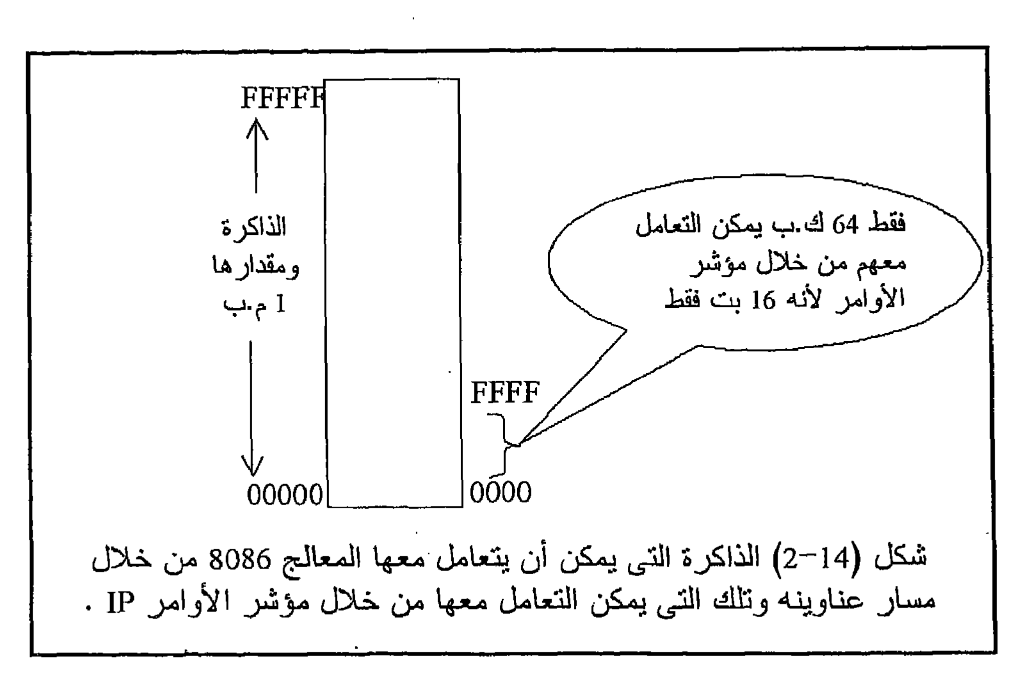
14-4 تجزىء الذاكرة Memory segmentation

مسار العناوين فى المعالج 8086/8088 يحتوى 20 بت أو بمعنى آخر يتكون من 20 خطا وهذا يعنى أن هذا المعالج يستطيع التعامل مع ذاكرة مقدارها 2^{20} أو 1048576 بايت أو اختصارا تكتب 1 ميجابايت (1 م.ب) . هذا يعنى أننا نستطيع مثلا أن نكتب أى برنامج فى خلال هذا المدى من الذاكرة والذى يبلغ 1 م.ب كما ذكرنا وعلى المعالج أن يحضر أوامر هذا البرنامج من الذاكرة وينفذها بالتتابع بلا أدنى مشاكل . إن هناك مشكلة صعبة تعوق المعالج من إمكانية إحضار الأوامر من الذاكرة بهذه السهولة وذلك لأن عداد البرنامج أو مؤشر الأوامر IP كما أسميناه هنا يحتوى على 16 بت فقط ، وكما نعلم أن مهمة المسجل IP هى أنه يحتوى عنوان الأمر الذى عليه الدور فى التنفيذ ، وهذا يعنى بالتالى أن أى أمر يقع فى الذاكرة خارج المدى العنوانى صفر إلى 64 ك.ب لن يستطيع المعالج إحضاره لأن مؤشر الأوامر IP يتكون من 16 بت فقط مما يعنى أن المدى العنوانى الذى يستطيع المعالج التعامل معه من خلال هذا المسجل هو صفر إلى 2^{16} أى 65536 بايت فى الذاكرة . فما هو الحل لهذه المشكلة ونحن نريد كتابة البرامج فى أى مكان فى الذاكرة التى يبلغ مداها 1 م.ب وليس فقط فى أول 64 ك.ب؟ شكل (14-2) يبين المدى العنوانى للمعالج 8086 على ضوء

عدد خطوط مسار العناوين ، والمدى العنوانى الذى يمكن التعامل معه من خـلال المسجل IP .

إن حل هذه المشكلة جاء من خلال استخدام فكرة زكية تمكنـك كمـبرمج مـن التعامل مع كل المدى العنوانى للذاكرة الذى يبلغ 1 م.ب بالرغم مـن اسـتعمال مسجلات 16 بت فقط وكان ذلك من خلال اسـتخدام أربـع مسـجلات سـميت بمسجلات تجزىء الذاكرة memory segmentation registers وكل منها 16 بـت ويرمز لها بالرموز التالية ES, SS, DS, CS وهذه الرموز لها دلالات تتطابق مـع وظيفة كل مسجل سنعرفها بعد قليل .

كل واحد من هذه المسجلات ، مسجلات التجزيء ، يحتوى عنوان من 16 بـت ولكن الظريف هنا أن هذه 16 بت تقابل أعلى 16 بت من مسار العنـاوين أى A4 إلى A19 وليس أول 16 بت A0 إلى A15 . أى أن محتويات أى واحد من هـذه المسجلات لن تمثل عنوانا حقيقيا فى الذاكرة إلا بعد إزاحتها ناحية اليسار بمقـدار 4بت أى بمقدار خانة ستعشرية أو بضربها فى الرقم 16 ضربا عشريا للحصـول على عنوان من 20 بت .



شكل (14-2) الذاكرة التى يمكن أن يتعامل معها المعالج 8086 من خلال مسار عناوينه وتلك التى يمكن التعامل معها من خلال مؤشر الأوامر IP .

فمثلا بافتراض أن المسجل C S محتوياته كالتالى : CS=0800H فـإن العنـوان الفعلى المقابل لهذه المحتويات هو 08000H بإضافة 0H ناحية اليمين أى بإزاحـة الرقم 4 بتات ناحية اليسار أو بضربه فى الرقم 16 ضربا عشـريا . وكذلـك إذا كانت محتويات المسجل DS كالتالى : DS=12F5H فإن العنوان الفعلى المقـابل

لهذه المحتويات هو 12F50H فى الذاكرة . هنا يظهر سؤال مهم وهو كيف يتم إحضار الأوامر من الذاكرة باستخدام مؤشر الأوامر IP الذى يتكون هو الآخر من 16 بت فقط ؟ وهل هذا المسجل له علاقة بمسجلات التجزىء ؟
بفرض أن محتويات مسجل التجزىء CS هى CS=12F0H ، وأن محتويات مؤشر الأوامر IP هى IP=001BH فما هو العنوان الحقيقى فى الذاكرة للأمر الذى عليه الدور فى التنفيذ ؟ يتحدد هذا العنوان بعد أن يقوم المعالج بإجراء الخطوتين التاليتين :

1-محتويات مسجل التجزىء CS تتم إزاحتها ناحية اليسار بمقدار 4 بت فتصبح المحتويات الجديدة هى :

CS=12F00H

2-تجمع محتويات مؤشر الأوامر مع محتويات المسجل CS بعد الإزاحة فيتكون لدينا العنوان الحقيقى كالتالى :

$$\begin{array}{r} CS = 12F00H \\ IP = 001BH \; + \\ \hline 12F1BH \end{array}$$

العنوان الحقيقى فى الذاكرة هو 12F1BH

أى أن الأمر الذى عليه الدور فى التنفيذ سيكون موجودا فى الذاكرة فى العنوان 12F1BH(20بت) كما رأينا فى المثال السابق . من ذلك نفهم حقيقة مهمة جدا وهي أن مؤشر الأوامر يشير أو يحدد عنوان في الذاكرة منسوبا أو محسوبا بمحتويات المسجل CS . ولنضرب لذلك المثال التوضيحى التالى : افترض أن لدينا سيارة هنا فى القاهرة وأقصى ما تستطيع أن تفعله هذه السيارة هو السير فى دائرة نصف قطرها 5 كيلومتر لتوزيع الحليب مثلا ، هذه هى مقدرتها ! ... فهل تستطيع هذه السيارة أن توزع الحليب فى لندن ؟ نعم تستطيع إذا نقلناها إلى لندن بالطائرة ! إن هذه السيارة تقابل مؤشر الأوامر الذى يحتوى فقط 16 بت ولا يستطيع التعامل إلا مع 64 كيلو بايت فقط ولكن هذه 64 كيلو بايت تتحدد بدايتها بمحتويات المسجل CS بعد إزاحتها ، وبذلك فإن مؤشر الأوامر يستطيع جلب أى أمر من أى مكان فى الذاكرة التى تبلغ 1 ميجابايت بعد جمع محتوياته مع محتويات المسجل CS التى تمت إزاحتها لليسار ، تماما مثل السيارة التى تستطيع أن توزع الحليب فى أى مكان فى العالم بعد نقلها بالطائرة للمكان المطلوب . لذلك فإنه فى بداية أى برنامج لابد من تحميل المسجل CS بالعنوان الذى نرغب فى كتابة البرنامج ابتداء منه وهذا العنوان بالطبع يكون فى أى مكان خلال الذاكرة التى تبلغ 1 ميجابايت . هنا يظهر سؤال وهو: لماذا ارتبط مؤشر الأوامر IP بالمسجل CS بالذات ولم يرتبط بأى واحد آخر من مسجلات

التجزىء مثل المسجل DS أو SS مثلا ؟ إن ذلك يرجع إلى الوظيفة المحددة لكل واحد من هذه المسجلات والتى نبينها فيما يلى :

14-4-1 مسجل تجزىء البرامج Code Segment register, CS

يحتوى هذا المسجل عنوان بداية جزء من الذاكرة يبلغ 64 كيلو بايت يخصص لكتابة شفرات البرامج فيه فقط ، ولذلك فإن مؤشر الأوامر يرتبط دائما بهذا المسجل لأن مؤشر الأوامر يشير على عنوان الأمر الذى عليه الدور فى التنفيذ ولابد أن الأمر يقع فى هذا الجزء من الذاكرة ، ويتحدد العنوان الحقيقى للأمر كما ذكرنا بإضافة محتويات مؤشر الأوامر مع محتويات المسجل CS بعد إزاحتها 4 بت ناحية اليسار . يمكن أن تتغير محتويات المسجل CS فى أثناء تنفيذ البرنامج مع أوامر القفز أو النداء على البرامج الفرعية وذلك فى حالات خاصة سيأتى شرحها بعد ذلك .

14-4-2 مسجل تجزىء البيانات Data Segment register, DS

يحتوى هذا المسجل على عنوان بداية جزء من الذاكرة يبلغ 64 كيلو بايت أيضا وهذا الجزء يحتوى جميع البيانات التى يتعامل معها أو يحتاجها البرنامج ، تتم عنونة هذه البيانات بإضافة محتويات المسجل DS بعد إزاحتها لليسار 4 بت مع محتويات أى واحد من المسجلات DX أو BX أو SI أو DI كما سنرى عند دراستنا لطرق العنونة .

14-4-3 مسجل تجزىء المكدسة Stack Segment register, SS

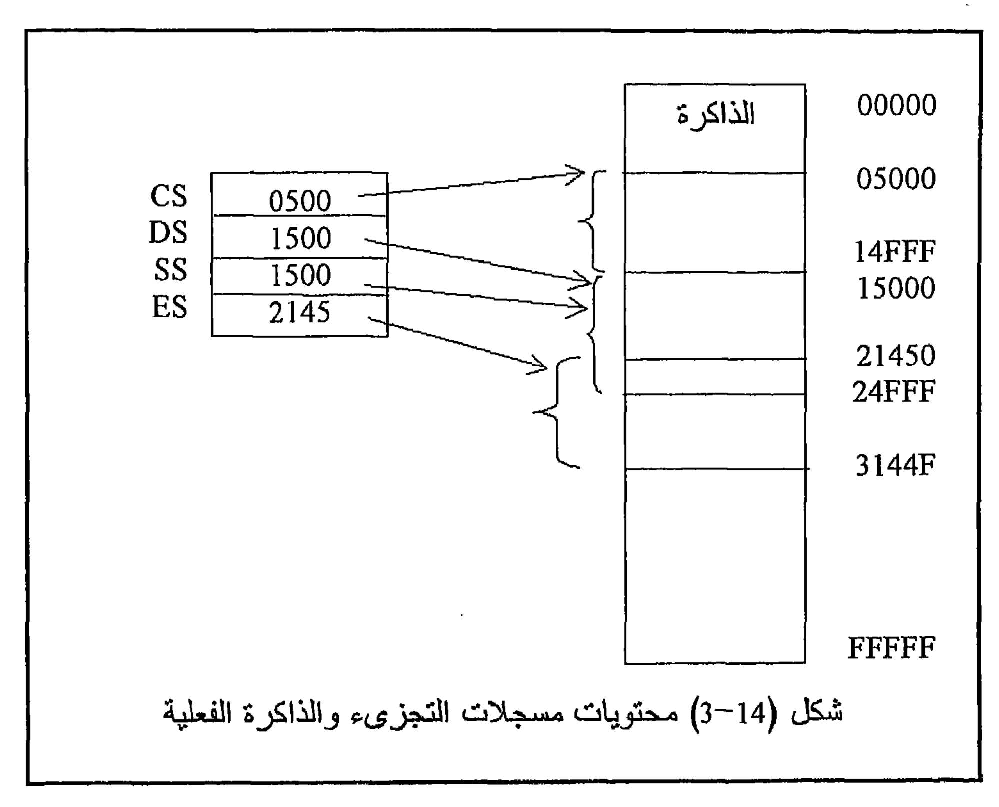
يحتوى هذا المسجل عنوان بداية جزء من الذاكرة يبلغ 64 كيلو بايت يستعملها المعالج كمكدسة . تستخدم المكدسة لتخزين البيانات والعناوين الضرورية عند القفز إلى البرامج الفرعية أو القفز إلى برنامج لخدمة مقاطعة حيث من شأن هذه البيانات والعناوين التى تخزن فى المكدسة أن تساعد المعالج على الرجوع إلى نفس المكان الذى تم القفز منه فى البرنامج الأساسى واسترجاع القيم الحقيقية لجميع المسجلات التى كانت موجودة قبل القفز بحيث يرجع المعالج إلى تنفيذ البرنامج الأساسى من حيث انتهى قبل القفز تماما دون خوف من تغير سير البرنامج بسبب فقد محتويات أحد المسجلات . يتم سحب البيانات من المكدسة على أساس أن آخر ما تم تسجيله يكون هو أول ما يتم سحبه أى أن آخر بايت تم تخزينها تكون أول بايت يتم سحبها Last In First Out, LIFO . تتم عملية السحب والإضافة فى المكدسة وبالتتابع الذى أشرنا إليه بمساعدة مسجل مؤشر المكدسة Stack Pointer, SP حيث يحتوى هذا المسجل الذى يتكون من 16 بت على عنوان آخر مكان فى المكدسة تم التخزين فيه وبالطبع فإن مقدار المكدسة يتحدد ب 64 كيلو بايت كما ذكرنا . يتم تحديد العنوان الفعلى أو الحقيقى داخل هذا الجزء من الذاكرة (المكدسة) عن طريق إزاحة محتويات مسجل تجزىء

المكدسة SS ناحية اليسار بمقدار 4 بتات ثم إضافة محتويات مؤشر المكدسة إليها.

14-4-4 مسجل التجزىء الاضافى Extra Segment register, ES

يحتوى هذا المسجل على عنوان بداية جزء من الذاكرة يبلغ 64 كيلو بايت تستخدم لتخزين البيانات أيضا وبالذات سلاسل الحروف strings . يتحدد العنوان الحقيقى أو الفعلى لأى معلومة داخل هذا الجزء بازاحة محتويات مسجل التجزىء ES ناحية اليسار بمقدار 4 بتات ثم يضاف إليه محتويات أى من المسجلين DI أو SI على حسب الأمر الذى يتم تنفيذه .

شكل (14-3) يبين محتويات مقترحة لكل واحد من مسجلات التجزىء والمساحة الفعلية التى يشغلها الجزء المقابل لكل مسجل على خريطة الذاكرة . نلاحظ من شكل (3ب) أن هذه الأجزاء يمكن أن تتداخل مع بعضها البعض ، بل ويمكن أن تشغل كلها نفس الجزء من خريطة الذاكرة .



شكل (14-3) محتويات مسجلات التجزىء والذاكرة الفعلية

14-4-5 مسجل الأعلام أو مسجل الحالة Status Register, SR

يحتوى هذا المسجل على 16 بت مستخدم منها 9 بتات فقط كأعلام وباقى بتات المسجل غير مستخدمه . إن كل بت أو علم من هذه الأعلام يعكس حالة معينة

من حالات نتيجة آخر عملية أو منطقية قام المعالج بتنفيذها . وفيما يلى نقدم هذه الأعلام والحالة التى يعكسها أو يبينها ومتى يكون كل علم صفرا ومتى يكون واحدا على ضوء هذه النتيجة (سبق شرح معظم هذه الأعلام ولكن لا مانع من مراجعة سريعة لوظائفها) .

1- علم الحمل Carry flag, CF

إذا حصل حمل أو استلاف من أو إلى آخر بت نتيجة إجراء أى عملية حسابية أو منطقية فإن علم الحمل يصبح واحدا ، ويكون صفرا فيما عدا ذلك . يتأثر هذا العلم أيضا ببعض أوامر الإزاحة والدوران .

2- علم الباريتى Parity flag, PF

إذا احتوت نتيجة آخر عملية أو منطقية نفذها المعالج على عدد زوجى من الوحايد فإن علم الباريتى يصبح واحدا ، أما إذا احتوت النتيجة على عدد فردى من الوحايد فإن هذا العلم يصبح صفرا .

3-علم الحمل النصفى Half carry flag, HF

يكون هذا العلم واحدا إذا حصل هناك حمل أو استلاف من أو إلى البت الثالثة (منتصف البايت) عند إجراء أى عملية حسابية أو منطقية . لاحظ أننا نعد البتات فى أى بايت ابتداء من الصفر ، أى البت رقم صفر ورقم واحد وهكذا .

4-علم الصفر Zero flag, ZF

يكون هذا العلم واحدا إذا كانت نتيجة آخر عملية حسابية أو منطقية نفذها المعالج تساوى صفرا ، ويكون هذا العلم صفرا إذا كانت النتيجة تختلف عن الصفر .

5-علم الإشارة Sign flag, SF

يوضح هذا العلم إشارة نتيجة آخر عملية حسابية أو منطقية نفذها المعالج ، فإذا كانت هذه النتيجة سالبة يكون هذا العلم واحدا وإذا كانت هذه النتيجة موجبة فإن هذا العلم يكون صفرا . لاحظ أن المعالج يعتبر النتيجة سالبة إذا كانت آخر بت فيها تساوى واحدا ويعتبر النتيجة موجبة إذا كانت آخر بت تساوى صفرا . من ذلك نقول أن علم الإشارة يساوى دائما آخر بت فى النتيجة .

6-علم الفخ أو المصيدة Trap flag, TF

هذا العلم لا يعكس نتيجة عملية نفذها المعالج ولكنه حينما يكون واحد فإن المعالج ينفذ البرنامج خطوة بخطوة وحينما يكون صفر فإنه ينفذها بالطريقة المعتادة .

7-علم تنشيط المقاطعة Interrupt enable flag, IF

لكى يمكن مقاطعة المعالج 8088/8086 فإنه يتم إعطاؤه إشارة على طرف طلب المقاطعة INTR (الطرف 18 فى شريحة المعالج) ولكن هذه المقاطعة لن يقبلها المعالج إلا إذا كان علم المقاطعة IF فعال أى يساوى واحدا هو الآخر .

8-علم الاتجاه Direction flag, DF

هناك بعض الأوامر الخاصة بالتعامل مع سلاسل الحروف character strings من خلال المسجلين SI, DI حيث يزاد واحد أو ينقص واحد من محتويات هذين المسجلين ليشير إلى مكان معين في هذه السلسلة . يبين علم الاتجاه DF إذا كان سيكون هناك زيادة أم سيكون هناك نقص بمقدار واحد على محتويات هذين المسجلين . إذا كان DF=1 فإن ذلك يعنى أنه سيكون هناك زيادة بمقدار واحد على محتويات هذين المسجلين ، وبالطبع إذا كان DF=0 فإن ذلك يعنى أنه سيكون هناك إنقاص بمقدار واحد على هذه المحتويات .

9-علم الفيضان Overflow flag, OF

يبين هذا العلم إذا كان هناك فيضان حسابى في نتيجة أى عملية حسابية مثل الجمع والطرح أم لا . فمثلا فى حالة جمع الرقم 7FH الذى يساوى (127+) مع الرقم (1H) فإن النتيجة تكون 80H والتى تعتبر سالبة بعد أخذ المتمم الثنائى لها وتساوى (128-) ، ولأن الرقم (128-) يعتبر غير صحيح لأن أى رقم سالب يجب ألا يتعدى (127-) فإن علم الفيضان يكون واحد . وعلى ذلك فإن هذا العلم يكون صفرا طالما لم يكن هناك فيضان في النتيجة .

7FH	127
+01H	001 +
80H	128-

14-5 طرق العنونة
Addressing modes

فى أثناء تنفيذ المعالج لأى برنامج فإنه ينقل بيانات من مسجل إلى مسجل آخر أو من مسجل إلى مكان ما فى الذاكرة أو من مكان ما فى الذاكرة إلى أى مسجل داخل المعالج نفسه . هناك طرق مختلفة يمكن استخدامها لكى يتم ذلك وهذه الطرق المختلفة يجب أن يلم بها أى مبرمج حتى يكون برنامجه ذو كفاءة عالية . سنقدم فى هذا الجزء شرحا مفصلا لهذه الطرق المختلفة من خلال استخدام الأمر MOV كمثال تطبيقى . يقوم الأمر MOV بنقل معلومة من مكان (المصدر) وهذا المصدر يكون إما مسجل داخل المعالج نفسه أو بايت من بايتات الذاكرة إلى مكان آخر (الهدف) وهذا الهدف أيضا يكون إما مسجل أو مكان فى الذاكرة . الصورة العامة لهذا الأمر هى :

```
MOV destination, source
```

حيث تنتقل المعلومة من المصدر إلى الهدف وكمثال على ذلك الأمر التالى :

MOV AX,BX

حيث تنتقل محتويات المسجل BX إلى المسجل AX وهو المركم . نلاحظ أنه دائما يكتب مصدر المعلومة بعد الفاصلة من ناحية اليمين وأما الهدف الذى ستنتقل إليه المعلومة فيكتب بجانب الأمر MOV وقبل الفاصلة .

14-5-1 عنونة المسجل Register addressing mode

تستخدم هذه الطريقة لنقل معلومة (بايت أو كلمة ، والكلمة 2 بايت) من مسجل إلى مسجل آخر ، أى أن مصدر المعلومة يكون مسجلا وكذلك الهدف . وهذه الطريقة تعتبر أسرع الطرق لنقل معلومة من مكان إلى مكان حيث كل من مصدر وهدف المعلومة يكون مسجلا داخل المعالج نفسه ولا يتعامل المعالج مع الذاكرة على الإطلاق . كمثال على ذلك الأمران :

MOV CX,AX

الذى ينقل محتويات المسجل AX (16بت) إلى المسجل CX (16بت أيضا) .

MOV AH,AL

الذى ينقل محتويات النصف الأول AL من المسجل AX إلى النصف الثانى AH فى المسجل نفسه .

من المهم جدا هنا أن نلاحظ أحجام المسجلات التى نتعامل معها ، فلا يصح مثلا أن ننقل محتويات مسجل 8 بت إلى مسجل 16بت أو العكس حيث سيعطى الأسمبلر رسالة خطأ على ذلك لأن ذلك غير مسموح . الجدير بالذكر هنا أنه فى مثل هذه الأوامر فإن مسجل المصدر لا تتغير محتوياته ولكن يؤخذ منها نسخة أو صورة وتوضع فى المسجل الهدف . فالأمر MOV CX,AX مثلا يأخذ نسخة من محتويات المسجل AX ويضعها فى المسجل CX دون تغير فى محتويات المسجل المصدر AX والذى يتغير فقط هو المسجل الهدف CX .

14-5-2 العنونة الفورية Immediate addressing mode

تستخدم هذه الطريقة لنقل معلومة (بايت أو كلمة) موجودة فى الأمر نفسه إلى مسجل من المسجلات . أى أن مصدر المعلومة هنا ليس مسجلا داخل المعالج ولا بايت فى الذاكرة ولكن المعلومة تعتبر ثابت أو قيمة موجودة فى الأمر نفسه ويعد شفرة الأمر مباشرة ، كمثال على ذلك الأمر :

MOV AX, 34F6H

الذى يضع نسخة من الثابت أو الرقم أو المعلومة 34F6H المكونة من 16بت والموجودة فى البرنامج بعد شفرة الأمر MOV فى المسجل AX . لاحظ أن H فى آخر أى رقم تعنى أن هذا الرقم مكتوبا فى النظام الستعشرى . بعض الأسمبلر تضع العلامة # أمام الثابت أو المعلومة الفورية ولكنها قليلة ونحن فى

هذا الكتاب لن نتبع ذلك وسنضع أى ثابت بدون هذه العلامة ، فقط سنضع حـروف H للدلالة على أن الرقم ستعشرى أو إذا كان الرقم فى النظام العشرى فلن نضـــع أى علامة .

3-5-14 العنونة المباشرة Direct addressing mode

هنا يتعامل المعالج مع الذاكرة حيث سيرسل لها أو يستقبل منها معلومة ، وعلـــى ذلك لابد من تحديد عنوان هذه المعلومة . فى العنونة المباشرة يحتـــوى الأمـــر نفسه على العنوان المباشر للمعلومة أو الثابت المراد جلبه أو إرساله من أو إلـــى الذاكرة . تذكر جيدا أن هذا العنوان يحدد نسبة إلى محتويات مسجل التجـــزىء DS ، فمثلا الأمر MOV AL,[1234H] معناه نقل نسخة من محتويات العنـــوان 1234H فى جزء البيانات إلى المسجل AL . لاحظ أنـــه بفـــرض أن محتويـــات المسجل DS=2000H فإن العنوان الفعلى للمعلومة السابقة سيكون 21234H بعـــد إزاحة محتويات المسجل DS لليسار 4بت كما رأينا مسبقا فى حسابات العنـــاوين الفعلية . إذا كان العنوان الذى سيتم التعامل معه فى جزء البيانات سيتكرر كثـــيرا فى البرنامج فإنه يمكن فى أول البرنامج إعطاء رمزا لهذا العنوان ثم بعـــد ذلـــك يستخدم هذا الرمز للدلالة على هذا العنوان فى أى مكان فى البرنـــامج . فمثـــلا يمكن أن نرمز للعنوان 1234H فى المثال السابق بالرمز NUMBER باســـتخدام الأمر NUMBER EQU 1234H فى أول البرنامج ، ثم بعد ذلك نستخدم الرمـــز NUMBER للدلالة على هذا العنوان كما فى الأمر MOV AL,NUMBER .

4-5-14 العنونة غير المباشرة Indirect addressing

هذه الطريقة من العنونة تسمح بالتعامل مع بيانات موجودة فـــى الذاكـــرة حيـــث العنوان الذى سيتم التعامل معه فى هذه الحالة يكون موجودا فى أحد مســـجلات المعالج التالية: BX, BP, SI, DI . كمثال على ذلـــك افـــترض أن المســـجل BX يحتوى الرقم 1000H وطلبنا من المعالج تنفيذ الأمر التـــالى: MOV AX,[BX] . فى هذه الحالة سيقوم المعالج بإحضار نسخة مـــن محتويـــات العنـــوان 1000H (والذى يليه) ويضعها فـــى المســـجل AX . أى أن محتويـــات المســـجل BX الموضوع بين قوسين مربعين كما رأينا تمثل عنوان المعلومة وليـــس المعلومـــة نفسها ، ففى عدم وجود القوسين سينسخ المعالج محتويات المسجل BX ويضعـــها فى المسجل AX كما رأينا فى أول طرق العنونة (عنونة المســـجل) . يجـــب أن نؤكد هنا أن العنوان الفعلى للمعلومة يحسب منسوبا لمحتويات مسجل التجـــزىء DS بعد إزاحته ناحية اليسار 4 بتات كمـــا ذكرنـــا ســـالفا ، أى أنـــه إذا كـــانت محتويات المسجل DS=0100H فإن الأمر السابق سينســـخ محتويـــات العنـــوان 01000+1000=02000H والذى يليه ويضعها فى المسجل AX . لاحظ أيضـــا أن

المسجلات BX, SI, DI تعنون عناوين منسوبة إلى مسجل التجزىء DS بينما المسجل BP فيعنون عناوين منسوبة لمسجل التجزىء SS . كأمثلة على هذا النوع من العنونة انظر إلى الأوامر التالية :

```
MOV CX,[BX]
MOV [BP],BL
MOV [DI],AH
MOV [DI],[BX]  (خطأ)
```

حيث الأمر الأول سينقل محتويات عنوان (والذى يليه) فى جزء البيانات أو ذاكرة البيانات data segment المشار إليه بالمسجل BX إلى المسجل CX ، بينما الأمر الثانى سينقل محتويات النصف الأول من المسجل BX إلى عنوان مشار إليه بالمسجل BP ويقع فى جزء المكدسة . الأمر الثالث سينقل النصف العلوى من المسجل AX إلى عنوان مشار إليه بالمسجل DI ويقع فى جزء البيانات ، أما الأمر الرابع فغير مسموح به لأنه ينقل من ذاكرة إلى ذاكرة وهذا النوع من العنونة غير مسموح به إلا فى حالات خاصة جدا مع بعض أوامر سلاسل الحروف .

14-5-5 عنونة القاعدة زائد الفهرسة Base plus index addressing

تعتبر هذه الطريقة من العنونة بمثابة عنونة غير مباشرة ولكن طريقة تكوين أو الحصول على العنوان تختلف عن الطريقة السابقة . هنا المسجلين BX و BP يستخدمان كقاعدة base أو كبداية لمجموعة أو صف أو مرصوصة array من العناوين حيث BX تستخدم فى حالة وجود مرصوصة البيانات فى جزء البيانات و BP تستخدم فى حالة وجود مرصوصة البيانات فى جزء المكدسة . فى هذا النوع من العنونة يتكون العنوان بجمع محتويات واحدة من مسجلات القاعدة BX أو BP مع محتويات واحد من مسجلات الفهرسة SI أو DI . يوضح ذلك المثال التالى :

```
MOV DL,[BX+DI]
```

حيث سينسخ المعالج محتويات العنوان المكون من جمع محتويات المسجلين BX و DI ويضعها فى النصف الأول من المسجل DX .

14-5-6 العنونة النسبية Relative addressing mode

هذا النوع من العنونة يختلف اختلافا بسيطا عن عنونة القاعدة زائد الفهرسة الذى تم شرحه سابقا حيث هنا يتم تحديد عنوان الذاكرة المراد التعامل معه عن طريق

جمع محتويات أحد المسجلات BX, BP, SI, DI مع إزاحة تعطى فى الأمر نفسه كما فى المثال التالى :

```
MOV AX,[BX+1000]
```

حيث هنا سيتم عنونة العنوان المحدد بجمع محتويات المسجل BX مع الرقم 1000H وهذا العنوان سيكون فى جزء البيانات من الذاكرة المحدد بمحتويات المسجل DS . جدول 14-1 يبين طرق العنونة السابقة مع مثال لكل طريقة ، حيث يمكنك مراجعته على ضوء ما سبق وبتأنى حتى يمكنك فهم هذه الطرق .

14-6 تمارين

1. ما هى المسجلات ذات 8 بت التى يمكن التعامل معها من خلال الأوامر للمعالج 8086/8088 ؟
2. قارن بين المعالجات 4 و 8 و 16 و 32 من حيث سرعة التنفيذ إذا تساوت كل العوامل الأخرى ؟
3. ما هى وحدة التنفيذ ووحدة مواجهة المسارات فى المعالجين 8086/8088 ؟ وما أثرهما على أداء المعالج ؟ وما هو الفرق بين كل وحدة فى كل من المعالجين ؟
4. ما هو طابور الإحضار Prefetch Queue ؟ وما هو تأثيره على أداء المعالج؟ وكم عدد البايتات فيها فى كل من المعالج 8086 و 8088 ؟
5. ما هى المسجلات ذات 16بت ، والمسجلات ذات 8 بت التى يمكن التعامل معها من خلال الأوامر للمعالج 8086/8088 ؟
6. لماذا يطلق على المسجل CX مسجل العد Count register ؟ والمسجل DX مسجل البيانات Data register ؟
7. ما هو الخطأ فى أوامر الانتقال التالية:

```
MOV AL,BX
MOV CS,SS
MOV ES,F214H
MOV [BX],[DI]
```

العنـــوان الفعلى للمعلومة فى الذاكـــرة	الأمــــــــــــــــــــــــــر
عنونة مسجل	MOV AL,BL
(10xDS)+temp	MOV AL, temp
عنونة فوريةImmediate	MOV AL,55H
(10xSS) + BP	MOV AL,[BP]
(10xDS) + BX	MOV AL,[BX]
(10xDS) + DI	MOV AL,[DI]
(10xDS) + SI	MOV AL,[SI]
(10xSS) + BP + 5	MOV AL,[BP+5]
(10xDS) + BX + 4H	MOV Al,[BX+4H]
(10xDS) + DI - 66H	MOV AL,[DI-66H]
(10xDS) + SI - 400H	MOV AL,[SI-400H]
(10xDS) + temp + BX	MOV AL, temp[BX]
(10xSS) + temp + BP	MOV AL, temp[BP]
(10xDS) + temp + SI	MOV AL, temp[SI]
(10xDS) + temp + DI	MOV AL, temp[DI]
(10xDS) + temp + BX + 10H	MOV AL, temp[BX+10H]
(10xSS) + temp + BP - 23H	MOV AL, temp[BP-23H]
(10xDS) + temp + SI + 50H	MOV AL, temp[SI+50H]
(10xDS) + temp + DI + 80H	MOV AL, temp[DI+80H]
(10xDS) + DI + BX	MOV AL,[BX+DI]
(10xSS) + DI + BP	MOV AL,[BP+DI]
(10xDS) + SI + BX	MOV AL,[BX+SI]
(10xSS) + SI + BP	MOV AL,[BP+SI]
(10xDS) + DI + BX + 8	MOV AL,[BX+DI+8]
(10xSS) + DI + BP - 10H	MOV AL,[BP+DI-10H]
(10xDS) + SI + BX -7	MOV AL,[BX+SI-7]
(10xSS) + SI + BP + 10H	MOV AL,[BP+SI+10H]
(10xDS) + temp + BX + SI	MOV AL, temp[BX+SI]
(10xSS) + temp + BP + SI	MOV AL, temp[BP+SI]
(10xDS) + temp + BX +DI	MOV AL, temp[BX+DI]
(10xSS) + temp + BP + DI	MOV AL, temp[BP+DI]
(10xDS) + temp + BX +SI +9	MOV AL, temp[BX+SI+9]
(10xSS) + temp + BP + SI - 10H	MOV AL, temp[BP+SI-10H]
(10xDS) + temp + BX +DI +200H	MOV AL,temp[BX+DI+200H]
(10xSS) + temp + BP + DI +1FH	MOV AL, temp[BP+DI+1FH]

جدول 14-1

8. أكتب أوامر الانتقال التى تقوم بالآتى:

- تحميل المسجل BX بالمعلومة 0F42H
- تحميل العنوان 32000H بالمعلومة 0BH
- تصفير بايتات الذاكرة (أى جعل محتوياتها تساوى صفـــرا) ابتـــداء مـــن العنوان 32000H إلى 32050H بالتتابع
- نقل محتويات الذاكـــرة 32000H حتـــى 32050H إلـــى 42000H حتـــى 42050H
- تحميل المسجلات CS, SS, DS, ES بالعنوان 3200H

9. ما معنى وضع القوسين [ ] حول أى معامل من معاملات أى أمر ؟

10. ما هو عنوان الذاكرة الذى سيتم التعامل معه فى كل من الأوامـــر التاليـــة إذا كـــانت DS=3200H, BX=0200H, DI=0300H, BP=1000H, SS=2000H و list=0250H :

- MOV AL,[3200H]
- MOV AL,[BX]
- MOV [DI],AL
- MOV AL,[BX+100]
- MOV AL,[BP+100H]
- MOV AL,[BP+DI]
- MOV AL, list[DI]
- MOV AL, list[100H]
- MOV AL,[BX+DI]

الفصل الخامس عشر

برمجة المعالج
Intel 8086/8088
والمصحح *Debugger*

15-1 مقدمة

سنرى في هذا الفصل الخطوات الأولى في اتجاه كتابة برنـــامج بســيط بلغـــة الأسمبلى الخاصة بالمعالج 8086/8088 ، ومن ثم تنفيذه ، كل ذلك مــــن خــلال برامج بسيطة نقدمها فقط لنفهم منها مكونات برامج لغة الأسمبلى للمعالج 8086 . بعد ذلك يعرض هذا الفصل لمجموعات أوامر هذه اللغة عرضا سريعا الغــرض منه هو التعريف بأهم مفردات هذه اللغة . بالطبع ســيتبقى هنـــاك الكثــير مــن الأوامر الأقل شيوعا ولكنها قد تفيد فى الكثير من التطبيقات ولكننا لن نتعــرض لها هنا ونحيل القارئ إلى أحد الكتب المتخصصة فى لغة التجميع المذكورة فـــى قائمة المراجع فى نهاية هذا الكتاب . إن الأمور عادة لا تأتي بكـــل مـــا يتمنـــاه المبرمج ، حيث كثيرا ما نجد أن البرنامج يحوي العديد من الأخطاء التى تعـــوق تنفيذ البرنامج بالصورة المطلوبة . سنرى في هذا الفصل أيضا بـــإذن الله كيفيـــة استخدام برنامج debugger أو المصحح "ديبجـــر" لاســتخراج هــذه الأخطـــاء والتخلص منها .

15-2 خطوات كتابة وتنفيذ برامج لغة التجميع

1. نبدأ بكتابة برنامج لغة الأسمبلى مستخدمين الأوامر المختلفة لهذه اللغة كمـــا سنرى تباعا بعد ذلك . يجب أن يتضمن البرنـــامج بعــض الأوامـــر الموجهـــة للأسمبلر لإخباره عن المتطلبات التي يحتاجها الأسمبلر عند تحويل البرنامج إلـــي لغة الماكينة . وأول هذه الأوامر هو أمر إخبار الأســمبلر عــن مكـــان وضــع البرنامج في الذاكرة مثلا ، وأيضا عن مكـــان وضــع البيانـــات الناتجـــة عـــن البرنامج.

بعد الانتهاء من كتابة البرنامج يجب أن يخزن في ملف file بأي أسم مع مراعـــاة أن يكون امتداد هذا الملف ASM. ، فمثلا يمكن تسمية الملف بأي واحـــد مـــن الأسماء التالية :

Example.ASM
Test.ASM

2. بعد ذلك يتم استدعاء الأسمبلر وإدخال الملف الذي تمت كتابته في الخطـــوة 1 عليه ، حيث سيعطينا الأسمبلر نتيجة ذلك ملف جديد بنفس الاسم السابق ولكـــن بامتداد مختلف ؛ هذا الملف سنسميه ملف الهدف object file وسيكون كالتالي :

example.obj
Test.obj

هذه الصورة من البرنامج تكون مكتوبة في صورة لغة الآلة ، ولكنها ما زالت غير مناسبة للتنفيذ بواسطة المعالج .

3. يتم بعد ذلك إدخال الملف السابق "ملف الهدف" على البرنامج الرابط linker الذي يقوم بتجميع الأجزاء المختلفة للبرنامج ، ووضعه في صورة مناسبة قابلة للتنفيذ executable بواسطة المعالج . هذه الصورة الجديدة للملف ستكون بنفس الاسم ولكن بامتداد جديد وهو EXE. وذلك كما يلي :

Example .EXE
Test .EXE

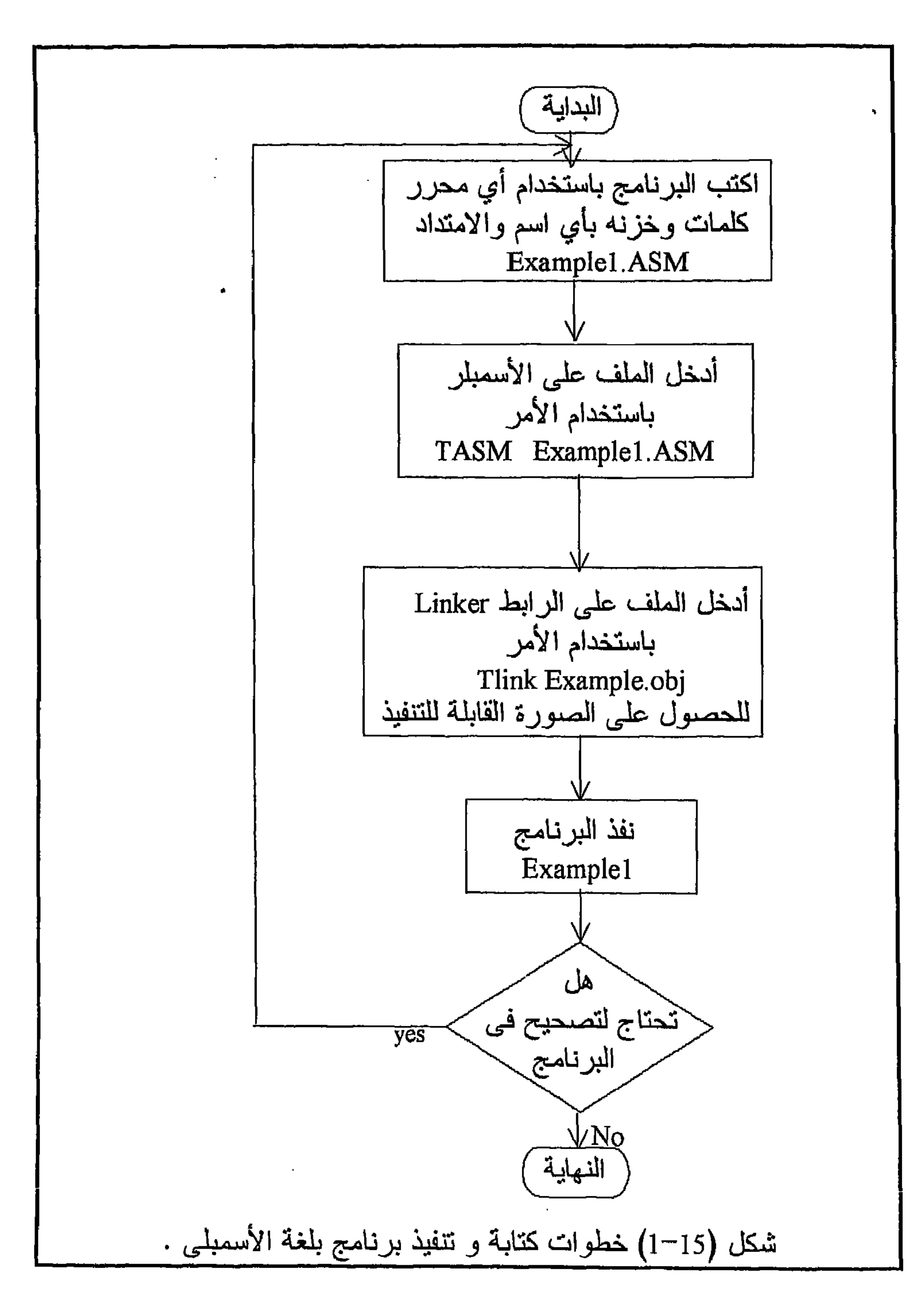
بعد الانتهاء من الخطوات الثلاث السابقة يمكن تنفيذ البرنامج ، وكذلك يمكن رؤية خرجه ، فإذا كان الخرج على ما يرام . . نكون قد انتهينا من البرنامج ، أما إذا جاءت النتائج على خلاف ما نتوقع ، فإن ذلك يدل على وجود أخطاء في البرنامج . . فكيف يمكننا إذن الكشف عن هذه الأخطاء والتعامل معها ؟ إن هذا يتم عن طريق استخدام برنامج الديبجر ، وسنرى في هذا الفصل كيفية الدخول في هذا البرنامج واستخدامه . شكل (15-1) يبين رسما توضيحيا لكتابة برنامج بلغة الأسمبلى ، وخطوات تنفيذه ، وذلك بفرض أن البرنامج تمت كتابته في ملف اسمه Example1.asm .

15-3 مكونات برنامج الأسمبلى

لكي تتعرف على مكونات برنامج لغة الأسمبلى ، سنسوق المثال الأول الذي يكتب الرسالة الآتية " آهلا يا عرب ، استيقظوا " على الشاشة ، وذلك دون أن ننشغل بتفاصيل البرنامج الآن ، حيث سيرد ذكرها فيما بعد بإذن الله .

مثال 15-1

```
DOSSEG
.MODEL SMALL
.STACK 100H
.DATA
message DB ' Hello Arab , woke up ',B,10,'$'
.CODE
mov ax, @data
mov ds,ax ; set ds to the beginning of the data segment
```



شكل (15-1) خطوات كتابة و تنفيذ برنامج بلغة الأسمبلى .

mov ah,9 ; load register ah evith 9

```
mov dx , offset message
int 21h
mov ah , 4ch
int 21h
END
```

من هذا المثال نرى الآتي :

1. وجود مجموعة من الأوامر في أول البرنامج وهي عبارة عن أوامر توجيهية للأسمبلر Assembler directives . هذه الأوامر تخبر الأسمبلر عن حقائق معينة يريد المبرمج أن يأخذها في الاعتبار ، مثل تحديد جزء ذاكرة البيانات data segment ، وجزء ذاكرة البرنامج code segment ، وجزء ذاكرة المكدسة stack segment وكذلك نهاية البرنامج .
2. القسم الثاني من الأوامر هو أوامر لغة الأسمبلى مثل الأوامر mov و add و sub وغيرها ، وكلها عبارة عن أوامر سيقوم الأسمبلر بتحويلها إلي شفرات لغة الآلة و تخزينها في الذاكرة . لاحظ أن الأوامر التوجيهية التي سبق الإشارة إليها لا يتم تحويلها إلي شفرات لأنها ليس لها شفرات أصلا ، وذلك لأنها ليست أوامر قابلة للتنفيذ بواسطة المعالج ، ولكنها مجرد توجيهات للأسمبلر لا يراها المعالج . فيما يلي سنأخذ فكرة موجزة عن أوامر التوجيه الموجودة في البرنامج السابق ، وهي كما يلي :

1- أمر التوجيه DOSSEG

هذا الأمر هو أول ما يكتب في أي برنامج أسمبلي ، وهذا الأمر يجعل كل أجزاء البرنامج (البيانات و الكود) تتبع نظام الميكروسوفت في التجزيء ، وهذا النظام لا يعنينا هنا في شئ ، ولن ننظر في أية تفاصيل أخرى له طالما أن هذا الأمر يقوم بهذه المهمة . المهم هنا هو أن نبدأ البرنامج بهذا الأمر كما ذكرنا .

2- أمر التوجيه .MODEL

يحدد هذا الأمر للأسمبلر موديل الذاكرة الذي سيتم التعامل معه ، حيث تبعا لهذا الموديل سيتحدد ما إذا كانت البيانات التي سيتعامل معها المعالج قريبة ؛ بحيث يتم عنونتها ب16 بت فقط داخل الجزء الخاص بها ، أم بعيدة فيتم التعامل معها على أساس 32 بت ، 16 منها تحدد العنوان داخل الجزء و 16 أخرى تحدد مكان أو بداية هذا الجزء . وهناك أكثر من موديل للذاكرة يتعامل معها الأسمبلر كما يلي :

1- الموديل tiny :

في هذا الموديل يكون البرنامج والبيانات موجودة في نفس الجزء حيث الجزء يبلغ 64 ك ب .

2- الموديل small :

يوجد كود البرنامج في جزء أو مقطع (64 ك ب) وبيانات البرنامج في جزء آخر منفصل عن الأول ولكن كلا من البرنامج والبيانات لا يتعدى الجزء الموجود فيه .

3- الموديل compact :

يوجد كود البرنامج في جزء معين ، أما بيانات البرنامج فيمكن أن تشغل أكثر من جزء واحد ، ولذلك فإن التعامل مع البيانات في هذه الحالة يكون على أساس أنها بعيدة ويكتب في 32 بت (segment : offset) بالرغم من أن البيانات هنا تشغل أكثر من جزء إلا أنه غير مسموح فى هذا الموديل أن يكون لمصفوفة واحدة array أن تخرج خارج حدود هذا الجزء ، أي أن أي مصفوفة لا تتعدى 64 ك ب .

4- الموديل large :

هنا يمكن لكود البرنامج وبياناته أن يشغل كل منهما أكثر من جزء واحد ، وسيكون التعامل مع العناوين هنا على الأساس البعيد سواء في حالة كود البرنامج أو بياناته . هنا أيضا يجب أن لا يتعدى حجم أي مصفوفة 64 ك.ب .

5- الموديل huge :

وهو يشبه تماما الموديل large في الوقت الحالي .

معظم البرامج التي نتعامل معها سواء في هذا المقرر أو في الكثير من التطبيقات الأخرى ، يكون الموديل small مناسبا جدا لها حيث أن البرنامج يكون مخصصا له 64 ك.ب ، وكذلك 64 ك.ب للبيانات ، وهذا يعتبر كافي جدا لهذه التطبيقات . ويجب أن نستخدم هذا الموديل كلما أمكن إلا إذا كانت هناك ضرورة لغير ذلك لأن العنونة البعيدة الموجودة في الموديلات 3 ، 4 ، 5 تأخذ وقتا أطول في التنفيذ . وأخيرا يجب أن يوضع الأمر MODEL. قبل أوامر تحديد الأجزاء المختلفة stack. و data. و code.

3- أمر التوجيه stack.

هذا الأمر يحدد كمية الذاكرة التي سيستخدمها البرنامج كمكدسة . والمكدسة يخزن فيها البرنامج عناوين الرجوع عند النداء على البرامج الفرعية ، أو تنفيذ برامج خدمة المقاطعة . إن 200 كلمة تعتبر مناسبة جدا كمكدسة في الكثير من الأغراض ، حيث يتم تحديد هذه الكمية كما في الأمر التالي :

```
.stack 200h
```

4- أمر التوجيه .code

هذا الأمر يحدد بداية الجزء الذي سنكتب فيه شفرات أو كود البرنامج . لاحظ أن هذا الأمر ليس له معاملات تكتب بعده كما كان في الأمر stack 200h. ولكن هذا الجزء يبدأ بالأمر code. وينتهي بالأمر END ، وكمثال على ذلك ما يلي :

```
----------
-----------
 .code
 add ax,bx
 sub ax,bx
 mov cx,100
-------------
-------------
END
```

5- أمر التوجيه DATA.

هذا الأمر يحدد بداية جزء البيانات المستخدمة في البرنامج كما يلي :

```
--------------
--------------
.DATA
boundary    DW 100
counter     DW 2
message     DW  ' **  ERROR MESSAGE ** ' , '$'
-------------
------------
```

6- أمر التوجيه END

بهذا الأمر تتحدد نهاية البرنامج ، وبدون هذا الأمر يعطي الأسمبلر رسالة خطأ ، لأنه من الضروري أن ينتهي البرنامج بهذا الأمر .

<u>ملحوظة : إن نسيان بعض أوامر التوجيه السابقة يسبب خطأ ، وبعضها يسبب تحذير ، لذلك نؤكد على ضرورة الالتزام بها .</u>

شكل (2-15) يبين الصورة العامة لبرنامج أسمبلى وقد احتوى كل أوامر التوجيه السابقة .

```
DOSSEG
.MODEL  SMALL
.STACK   100H
.DATA
 DB
 DW
 ------------
 -------------
.Code
    Your program


END
```

شكل (15-2) الصورة العامة لبرنامج الأسمبلي .

15-4 أوامر لغة الأسمبلى

لغة الأسمبلى للشريحة 8088/8086 تحتوى العديد من الأوامر بحيث أنه سيكون من الصعب ومن الممل جدا أن ندرس هذه الأوامر عن طريق سردها الواحد بعد الآخر إلى أن نصل إلى نهايتها بحيث عندما نصل إلى النهاية نكون قد نسينا ما درسناه فى البداية . لذلك فقد اخترنا أن نقسم هذه الأوامر إلى مجموعات كما فعلنا عند دراسة لغة الأسمبلى للمعالجات السابقة بحيث ندرس كل مجموعة على حدة مع إعطاء بعض الأمثلة السريعة والتمارين على كل مجموعة ، معتمدين على أن الدارس لديه الخبرة الآن بمعظم أساسيات البرمجة بهذه اللغة .

15-5 مجموعة أوامر الانتقال
Transfer instructions

هذه المجموعة من الأوامر خاصة بنقل البيانات من مكان لآخر دون إجراء أى تعديل أو تغيير عليها . الصورة العامة لهذه الأوامر هى :

mov destination, source

حيث الكلمة mov هى اختصار لكلمة move بمعنى أنقل أو حرك ، وأما source فهو مصدر المعلومة ، و destination هو الهدف أو الملجأ الذى تذهب إليه المعلومة . أى أن المعلومة ستنتقل من المصدر إلى الهدف . كل من المصدر

والهدف من الممكن أن يكون مسجلا من مسجلات المعالج أو عنوان من عنـاوين الذاكرة . كما يمكن أن يكون المصدر معلومة فورية immediate أو ثابت .
من أمثلة نقل البيانات بين المسجلات المختلفة ما يلي :

```
mov  al,bl ;      نقل محتويات المسجل bl (8بت) إلى المسجل al(8بت)
mov  ax,cx ;      نقل محتويات المسجل cx (16بت) إلى المسجل ax (16بت)
mov  bp,sp ;      نقل محتويات المسجل sp (16بت) إلى المسجل bp (16بت)
mov  ds, ax
mov  di,si
mov  bx,es
mov cs,ds ;       هذا الأمر خطأ لأنه لا يمكن نقل محتويات مسجل مقطع إلى
           ;      إلى مسجل مقطع آخر
mov  bl,ax ;      هذا الأمر خطأ لأن المسجلين أحدهما 16 بت والآخر 8 بت
```

جميع الأوامر السابقة كانت تتعامل مع مسجلات فقط سواء كمصدر للمعلومـة أو هدف ستذهب إليه المعلومة . هذا هو ما يسمى أحيانا بعنونة المسجلات register addressing حيث لا يكون هناك تعامل مع الذاكرة فى طرفـي الأمـر ، فقـط مسجلات . يمكن تحميل أى مسجل بمعلومة فورية immediate data 8 بـت أو 16 بت كما فى الأوامر التالية :

```
mov  al,03h
```

هذا الأمر يحمل النصف الأول من مسجل التراكم (8 بت) بالمعلومة الفورية 03h (8 بت) ، حيث الحرف h يعنى أن هذه المعلومة مكتوبة فى النظام الستعشرى .

```
mov ax,0ff35h
```

حيث هنا تم تحميل المسجل ax (16بت) بالمعلومة ff35h (16بت) . عادة يوضـع صفر قبل أى رقم ستعشرى يبدأ بحرف كما فى المثال السابق .
من المفيد جدا فى الكثير من البرامج أن نرمز لقيمة فورية بأي رمز ثم نسـتخدم هذا الرمز فى البرنامج بدلا من القيمة الثابتة ، كما فى الأوامر التالية :

```
kkk   equ   33h
..................
mov  al,kkk
```

equ هو أمر توجيه جديد موجه للأسمبلر بإعطاء القيمة 33h للرمز kkk حيـث يقوم الأسمبلر باستبدال الرمز kkk بقيمته عند كل موضع يظهر فيه هذا الرمـز فى البرنامج .

مثال 15-2

أكتب برنامج يحمل المسجلات ah, al, bh, bl, ch, cl, dh بالقيم 00, 01, 02, 03, 04, 05, 06 على الترتيب ثم يقوم بعمل إزاحة دورانية على محتويات هذه المسجلات . هذا المثال تم تناوله مع كل المعالجات 8 بت ولذلك سنقدم البرنامج مباشرة كالتالى :

```
dosseg
.model small
.stack 100h
.data
.code
mov ah,00
mov al,01h
mov bh,02h
mov bl,03h
mov ch,04h
mov cl,05h
mov dh,06h
mov  dl.dh
mov  dh,cl
mov cl,ch
mov  ch,bl
mov  bl,bh
mov  bh,al
mov  al,ah
mov  ah,dl
end
```

بعد كتابة هذا البرنامج سجله فى ملف اسمه example1.asm وبعد ذلك استدعى الأسمبلر وأدخل عليه البرنامج باستخدام الأمر :

Tasm example1

للحصول على برنامج الهدف example1.obj . بعد ذلك استدعى برنامج التوصيل tlinker للحصول على الصورة القابلة للتنفيذ للبرنامج كما يلي :

tlink example1

بالحصول على الصورة القابلة للتنفيذ من البرنامج يمكنك تنفيذه باستخدام الأمر :

example1

حيث سينفذ البرنامج ويرجع الحاسب إلى dos دون أن ترى نتيجة محسوسة للبرنامج لأن مثل هذا البرنامج لا يطبع شيئا على الشاشة ولا يرسل نتائج إلى الطابعة ، لذلك فلن تحس به لأنه فقط يغير من محتويات المسجلات داخل

المعالج . فى مثل هذه الظروف يلعب الديبجر دورا مهما فى أنه يمكننا به أن نرى نتيجة تنفيذ البرنامج فى المسجلات ، حيث يمكن باستخدام الديبجر أن نفحص كل مسجلات المعالج لنرى محتوياتها بعد تنفيذ البرنامج لنعرف هل تم تنفيذ البرنامج بالطريقة المطلوبة أم لا . بل إنه من مزايا استخدام الديبجر أنه يمكننا تنفيذ البرنامج خطوة بخطوة لنرى نتيجة البرنامج بعد تنفيذ كل أمر ونفحص عند أى لحظة لنعرف هل البرنامج يسير على ما يرام أم لا . وهذه فى الحقيقة تعتبر فائدة عظيمة فى استخراج الأخطاء من البرامج . لذلك سنعرض فى الجزء القادم لكيفية الدخول فى البرنامج من الديبجر واستخدامه لتتبع تنفيذ البرنامج .

6-15 الديبجر Debugger

لكي تدخل فى الديبجر لابد وأن يكون لديك الصورة القابلة للتنفيذ من البرنامج الذى تريد استخراج أخطاؤه أو التعامل معه ، لذلك يمكننا الدخول فى الديبجر بالأمر التالي :

C:\TASM>debug example2.exe

بذلك تدخل فى الديبجر وتظهر لك علامة وجودك فيه حيث يصبح دليل الكتابة Cursor هو الشكل '-' ويمكنك استخدام أوامره كالتالي :

1-6-15 إظهار محتويات المسجلات بالأمر R

بكتابة الحرف R (أو r لأن لغة الأسمبلى ليست حساسة لشكل الحرف) ثم enter تظهر أمامك جميع المسجلات بمحتوياتها كما يلى :

```
-r
AX=0000  BX=0000  CX=001E  DX=0000  SP=0100  BP=0000  SI=0000  DI=000
DS=272C  ES=272C  SS=273E  CS=273C  IP=0000 NV UP EI PL NZ NA PO NC
```

حيث نرى أن محتويات المسجل AX=0000 والمسجل CS=273C وهكذا .
يمكنك إظهار محتويات مسجل معين بكتابة اسم المسجل بعد الحرف R حيث تظهر لك محتويات هذا المسجل فقط وفى السطر التالي تظهر العلامة ':' والتى تتيح لك تغيير محتويات هذا المسجل بكتابة المحتويات الجديدة بعد هذه العلامة ، وإذا لم تريد تغيير هذه المحتويات اضرب enter .
يظهر فى آخر قائمة المسجلات بيان بحالة جميع الأعلام flags الموجودة فى المعالج وحالة كل علم إذا كان واحد أم صفر . جدول (15-1) يبين قائمة بهذه

الأعلام وماذا يكتب فيها إذا كانت صفرا وماذا يكتب فيـــها إذا كـــانت واحـــد .
سندرس معنى هذه الأعلام بالتفصيل عند دراستنا للقفز المشروط .

اسم العلم	العلم مرفوع set to one	العلم غير مرفوع set to zero
Over flow	OV	NV
Direction	DN	UP
Interrupt	EI	DI
Sign	NG	PL
Zero	ZR	NZ
Auxiliary	AC	NA
Parity	PE	PO
Carry	CY	NC

جدول (15-1) بيان بحالة الأعلام التى يظهرها الديبجر

15-6-2 عرض أوامر الأسمبلى ابتداء من عنوان معين Un

تحتوى الذاكرة الشفرات الثنائية لأوامر البرنامج ، وعرض هذه الشفرات الثنائيـــة بنفس حالتها لا يفيد شئ حيث يكون من الصعب فهمها . لذلك فقد أتاح الديبجـــر إمكانية عرض هذه الأوامر بشفرات الأسمبلى عن طريق كتابة الأمر Un والـذى يعنى عرض n من الأوامر ابتداء من العنوان الموجود فى المسجل CS كما يلى:

```
C:\TASM>debug example2.exe
-r
AX=0000 BX=0000 CX=001E DX=0000 SP=0100 BP=0000 SI=0000 DI=0000
DS=1898 ES=1898 SS=18AA CS=18A8 IP=0000 NV UP EI PL NZ NA PO NC
18A8:0000 B400        MOV  AH,00
-u0
18A8:0000 B400        MOV    AH,00
18A8:0002 B001        MOV    AL,01
18A8:0004 B702        MOV    BH,02
18A8:0006 B303        MOV    BL,03
18A8:0008 B504        MOV    CH,04
18A8:000A B105        MOV    CL,05
18A8:000C B606        MOV    DH,06
18A8:000E 8AD6        MOV    DL, DH
18A8:0010 8AF1        MOV    DH, CL
18A8:0012 8ACD        MOV    CL, CH
18A8:0014 8AEB        MOV    CH, BL
18A8:0016 8ADF        MOV    BL, BH
18A8:0018 8AF8        MOV    BH, AL
18A8:001A 8AC4        MOV    AL, AH
18A8:001C 8AE2        MOV    AH, DL
```

```
18A8:001E 31E4        XOR    SP, SP
-
```

نلاحظ فيما سبق أن محتويات المسجل CS=18A8 ، لذلك تم عـــرض الأوامــر ابتداء من العنوان 0000 نسبة لمحتويات هذا المسجل . بعد العنوان مباشرة ستجد شفرة الأمر الستعشرية ، فمثلا الأمر MOV AH,00 كانت شفرته B400 والأمــر MOV AH,DL شفرته هى 8AE2 .

15-6-3 عرض محتويات جزء من الذاكرة بالشــفرات الستعشــرية بالأمر Dn

يمكن عرض محتويات جزء معين من الذاكرة بالأمر Dn حيث n هــى عنــوان البداية التى سيبدأ من عندها عرض المحتويات ، وعنــوان البدايــة يكــون هــو العنوان الموجود فى المسجل DS ، كما يلى :

```
C:\TASM>debug example2.exe
-r
AX=0000 BX=0000 CX=001E DX=0000 SP=0100 BP=0000 SI=0000 DI=0000
DS=1898 ES=1898 SS=18AA CS=18A8 IP=0000  NVUP EI PL NZ NA PO NC
18A8:0000 B400        MOV    AH,00
-D0
1898:0000 CD 20 00 A0 00 9A F0 FE-1D F0 4F 03 FD 11 8A  03............
1898:0010 FD 11 17 03 FD 11 EC 11-01 01 01 00 02 FF FF FF   ..............
1898:0020 FF FF FF FF FF FF FF FF-FF FF FF FF 83 18 4C 01
1898:0030 98 18-FF FF FF FF 00 00 00 00
-
```

لاحظ أن محتويات المسجل DS=1898 وتم عرض محتويــات أمــاكن الذاكــرة منسوبة لهذا العنوان وكل سطر يبين محتويات 16عنوان (10 ستعشرى) .

15-6-4 تنفيذ البرنامج حتى عنوان معين Ga

هذا الأمر يبدأ فى تنفيذ البرنامج ابتدأ من العنوان المحدد بمسجل التجــزيء CS ومحتويات مؤشر الأوامر IP . أى عنوان بداية التنفيذ سيكون CS:IP . ســيقف التنفيذ عند العنوان a الموجود بعد الحرف G بحيث لن يتم تنفيذ الأمر الموجــود عند هذا العنوان . لذلك لتنفيذ البرنامج السابق من بدايته لابد من تصفير المســجل IP أولا باستخدام الأمر RIP – ثم تحميل المسجل IP بأصفــار . بذلــك ســيبدأ التنفيذ من العنوان 1898:0000 . بعد ذلك نعطيه أمر التنفيــذ G001E – حيــث

سيتوقف التنفيذ عند هذا الأمر الذى لا يدخل ضمن أوامر البرنامج ولذلك فإنه لـن ينفذ .

15-6-5 متابعة تنفيذ البرنامج عن طريق تنفيذ عدد n من الخطوات Tn

يمكن متابعة Trace, T تنفيذ البرنامج لاستخراج أخطاء التنفيذ عن طريق تنفيـذه خطوة بخطوة باستخدام الأمر Tn حيث n هى عدد الأوامر المطلوب تنفيذهـا . فمثلا T1 ستنفذ خطوة واحدة من البرنامج ، وهكذا . تذكر هنا أيضا أن مؤشـر الأوامر IP لابد أن يحتوى عنوان الخطوة المراد تنفيذها منسوبا لمسجل التجزيء CS . بعد تنفيذ كل خطوة يظهر الديبجر محتويات جميع المسجلات وكذلك الأمـر التالي فى التنفيذ كما يلي :

```
C:\TASM>debug example2.exe
r-
AX=0000 BX=0000 CX=001E DX=0000 SP=0100 BP=0000 SI=0000 DI=0000
DS=1898 ES=1898 SS=18AA CS=18A8 IP=0000  NVUP EI PL NZ NA PO NC
18A8:0000 B400        MOV    AH,00
t1-
AX=0000 BX=0000 CX=001E DX=0000 SP=0100 BP=0000 SI=0000 DI=0000
DS=1898 ES=1898 SS=18AA CS=18A8 IP=0002  NVUP EI PL NZ NA PO NC
18A8:0002 B001        MOV    AL,01
t1-
AX=0001 BX=0000 CX=001E DX=0000 SP=0100 BP=0000 SI=0000 DI=0000
DS=1898 ES=1898 SS=18AA CS=18A8 IP=0004  NVUP EI PL NZ NA PO NC
18A8:0004 B702        MOV    BH,02
t2-
AX=0001 BX=0200 CX=001E DX=0000 SP=0100 BP=0000 SI=0000 DI=0000
DS=1898 ES=1898 SS=18AA CS=18A8 IP=0006 NVUP EI PL NZ NA PO NC
18A8:0006 B303        MOV    BL,03

AX=0001 BX=0203 CX=001E DX=0000 SP=0100 BP=0000 SI=0000 DI=0000
DS=1898 ES=1898 SS=18AA CS=18A8 IP=0008 NVUP EI PL NZ NA PO NC
18A8:0008 B504        MOV    CH,04
```

15-6-6 تغيير محتويات عنوان فى الذاكرة Ea

فى الكثير من الأحيان يتطلب تنفيذ البرنامج وضع قيمة معينة فى عنوان محـدد فى الذاكرة . يتم ذلك بالأمر Ea حيث a هـى عنـوان البـايت المـراد تغيـير محتوياتها ، حيث بعد كتابة هذا الأمر يعرض الديبجر محتويـات هـذا العنـوان الموجودة فعليا وينتظر منك تغيير هذه القيمة .

15-6-7 الخروج من الديبجر Q

بكتابة الحرف Q ثم enter تخرج من الديبجر إلى دوس .
حاول كتابة مثال 1 الخاص بالإزاحة الدورانية لمحتويات المسجلات وجرب عليه كل أوامر الديبجر .

15-7 تمـــاريـــن

1. أشرح ما هو المقصود بالعنونة الضمنية ؟
2. ما هو المقصود بعنونة المسجلات ؟
3. أكتب باختصار عن الديبجر ؟ ومـــاذا تفعـــل لكـــي تتبـــع تنفيـــذ البرنـــامج خطوة بخطوة ؟
4. أكتب برنامج يحمل المسجلين AH , AL بأي بيانـــات ثـــم يقـــوم البرنـــامج باستبدال محتويات هذين المسجلين دون فقد محتويات أى مسجل منهما؟
5. أعد التمرين السابق ولكن على مسجلين 16 بت ، AX , BX مثلا ؟
6. أعد برنامج الإزاحة الدورانية فى مثال 1 ولكن هذه المرة اجعـــل الإزاحـــة تكون ناحية اليسار بدلا من ناحية اليمين كما كان فى المثال ؟
7. تتبع تنفيذ البرنامج التالي مع كتابة محتويات المسجلات بعد تنفيذ كل خطـــوة باعتبار أن جميع المسجلات تحتوى أصفار فى بداية البرنامج :

	AH	AL	BH	BL	CH	CL	DH	DL
	00	00	00	00	00	00	00	00
MOV AL,05	--	--	--	--	--	--	--	--
MOV AH,01	--	--	--	--	--	--	--	--
MOV BX,AX	--	--	--	--	--	--	--	--
MOV CL,BH	--	--	--	--	--	--	--	--
MOV CH,BL	--	--	--	--	--	--	--	--
MOV DX,CX	--	--	--	--	--	--	--	--

15-8 أوامر القفز
Jump Instructions

القاعدة العامة أن المعالج يقوم بتنفيذ البرنامج حسب ترتيب الأوامر من أول البرنامج حتى نهايته ، ولقد راعينا ذلك فى الأمثلة السابقة . ولكن هناك بعض التطبيقات التى تتطلب الخروج على هذه القاعدة ، كأن يطلب منك مثلا تنفيذ عملية معينة أو مجموعة من الأوامر عدد معين من المرات أو حتى عدد لا نهائي من المرات . لقد أتاح المعالج ذلك بتوفير بعض الأوامر التى تمكنك كمبرمج من القفز بعملية التنفيذ من مكان لآخر خلال البرنامج . عادة تنقسم أوامر القفز إلى نوعين كالتالي :

15-8-1 القفز غير المشروط Unconditional jump

عند تنفيذ أوامر القفز غير المشروط ينتقل المعالج بعملية التنفيذ إلى المكان الجديد والمحدد دون قيد أو شرط . هذا المكان الذى سيتم القفز إليه يكون محددا بعلامة معينة label حيث تستخدم هذه العلامة فى أمر القفز كذلك . هناك أمر وحيد للقفز غير المشروط وهو الأمر :

```
jmp label
```

كمثال على ذلك ما يلى :

```
again: mov ax,05H
       mov bx,ax
       mov cx,bx
      jmp again
```

حيث العلامة again تم استخدامها قبل الأمر mov ax,05H المراد القفز إليه ، كما تم استخدامها أيضا فى أمر القفز نفسه . من شروط العلامات المستخدمة أنها لابد أن تبدأ بحرف أبجدي ومن الممكن أن تحتوى أرقاما ومن الممكن أن يصل عدد حروف العلامة إلى 31 حرفا . يجب أيضا ألا تحتوى العلامة على مسافات ، ويجب أن تنتهي بالحرف ":" كدليل يبين نهاية العلامة . كذلك لابد من وجود مسافة بين نهاية العلامة ":" وبداية الأمر .

15-8-2 القفز المشروط Conditional jump

كما يوحي الاسم فإن هذا النوع من القفز لن يتم إلا إذا تحقق شرطا معينا ، إذا لم يتحقق هذا الشرط فإن القفز لن يتم وسيستمر البرنامج فى مساره الطبيعي حيث ينفذ الأمر التالي لأمر القفز . إن شروط القفز توضع دائما على الأعلام ، فهناك

قفز مثلا إذا كانت النتيجة تساوى صفرا ، أى أن علم الصفر يساوى واحد ، كما أن هناك قفزا إذا كان هناك حملا فى آخر عملية قام بها المعالج ، وهكذا . جدول (2-15) يبين معظم أوامر القفز الشهيرة والكثيرة الاستخدام مع المعالج 8086/8088 . هناك أيضا أوامر قفز مشروطة بحالة معينة لأكثر من علم مثل الأمر JA والذى يعنى اقفز إذا كانت النتيجة أكبر من الصفر ، وتكون النتيجة أكبر من الصفر إذا كان علم الصفر يساوى صفرا وعلم الإشارة يساوى صفرا أيضا .

أمر القفز	وصف الأمر
JA	اقفز إذا كانت النتيجة فوق الصفر jump if above
JAE	اقفز إذا كانت النتيجة فوق الصفر أو تساوى صفر
JB	اقفز إذا كانت النتيجة تحت الصفر jump if below
JBE	اقفز إذا كانت النتيجة تحت الصفر أو تساويه
JE/JZ	اقفز إذا كانت النتيجة تساوى الصفر jump if equal
JNC	اقفز إذا لم يكن هناك حمل jump if no carry
JNE	اقفز إذا كانت النتيجة لا تساوى الصفر
JNO	اقفز إذا لم يكن هناك فيضان فى النتيجة
JPO	اقفز إذا كانت الباريتى فردية jump if parity odd
JNS	اقفز إذا كانت النتيجة موجبة jump if no sign
JO	اقفز إذا كان هناك فيضان فى النتيجة
JPE	اقفز إذا كانت الباريتى زوجية jump if parity even
JS	اقفز إذا كانت النتيجة سالبة jump if sign
JG	اقفز إذا كانت النتيجة أكبر من الصفر jump if greater
JGE	اقفز إذا كانت النتيجة أكبر من الصفر أو تساويه
JL	اقفز إذا كانت النتيجة أقل من الصفر jump if less
JLE	اقفز إذا كانت النتيجة أقل من أو تساوى
JCXZ	اقفز إذا كان المسجل CX=0 يساوى صفر

جدول (2-15) أوامر القفز المشروط

مثال 15-3

أكتب برنامجا يقارن محتويات المسجل AX والمسجل BX بحيث إذا كانا متساويان يضع 1 فى المسجل DX ، أما إذا كانا غير متساويين فيضع صفر فى المسجل DX هذا مع الحفاظ على محتويات كل من المسجلين AX و BX .

أحد الاقتراحات لهذا البرنامج هو البرنامج التالي مع العلم أنه من الممكن أن يكون هناك أكثر من حل بالذات بعد أن ندرس باقي أوامر الحساب .

```
        MOV CX,AX
        SUB AX,BX
        JZ HERE1
        MOV DX,0
        JMP HERE2
HERE1:  MOV DX,1
HERE2:  MOV AX,CX
```

نلاحظ أن هذا البرنامج قد احتفظ بمحتويات المسجل AX فى المسجل CX كما فى الأمر الأول ، بعد ذلك طرح محتويات المسجل BX من المسجل AX حيث ستوضع النتيجة فى المسجل AX ، لذلك فإن محتويات المسجل AX ستفقد ولذلك فقد احتفظنا بها فى المسجل CX . بعد ذلك إذا كانت نتيجة الطرح صفرا فإن المسجلين متساويين وسيضع البرنامج 1 فى المسجل DX بعد أن يقفز إلى العلامة HERE1 . أما إذا كان المسجلين غير متساويين فسيضع البرنامج صفر فى المسجل DX ثم يقفز إلى العلامة HERE2 ليسترد محتويات المسجل AX من المسجل CX .

15-8-3 الأمر LOOP

من الأوامر الكثيرة الاستخدام لعمل الحلقات الأمر LOOP والذى ينفذ مجموعة الأوامر المحصورة بينه وبين العلامة المذكورة فيه عدد من المرات يساوى محتويات المسجل CX . كمثال على ذلك انظر إلى ما يلي :

```
       MOV CX,10H
HERE:  MOV AX,00H
       ...........
       ..........
       LOOP HERE
```

حيث سينفذ هذا البرنامج مجموعة الأوامر الموجودة بين الأمر LOOP والعلامة HERE1 عدد 16 (10H) مرة حيث هذا العدد مخزن فى المسجل CX قبل الدخول فى البرنامج .

15-9 أول خطوات التعامل مع الذاكرة
First step to memory addressing

هناك طريقتان للتعامل مع الذاكرة وهما كما يلي :

15-9-1 الطريقة المباشرة Direct addressing

فى هذه الطريقة يوجد العنوان المراد التعامل معه فى الأمـــر نفســه مباشـــرة ، ولذلك سميت هذه الطريقة بالطريقة المباشرة للتعامل مع الذاكرة .
كمثال على ذلك الأوامر التالية :

MOV al,[2000H]

وهذا الأمر يعنى نقل محتويات البايت أو العنـــوان 2000H فـــى الذاكـــرة إلـــى المسجل al .

MOV AX,[2000H]

والذى يعنى نقل محتويات العنوان 2000H والذى يليه إلى المسجل AX .

MOV [31F2H],AL

والذى يعنى نقل محتويات المسجل AL إلى البايت التى عنوانها 31F2H .
نلاحظ أنه فى كل هذه الأوامر ظهر العنوان مباشرة فى الأمر نفسه ، ونلاحـــظ أيضا أن العنوان تم وضعه بين القوسين المربعين [ ] .
من الممكن أن يرمز للعنوان برمز معين كما فى الأمر التالي :

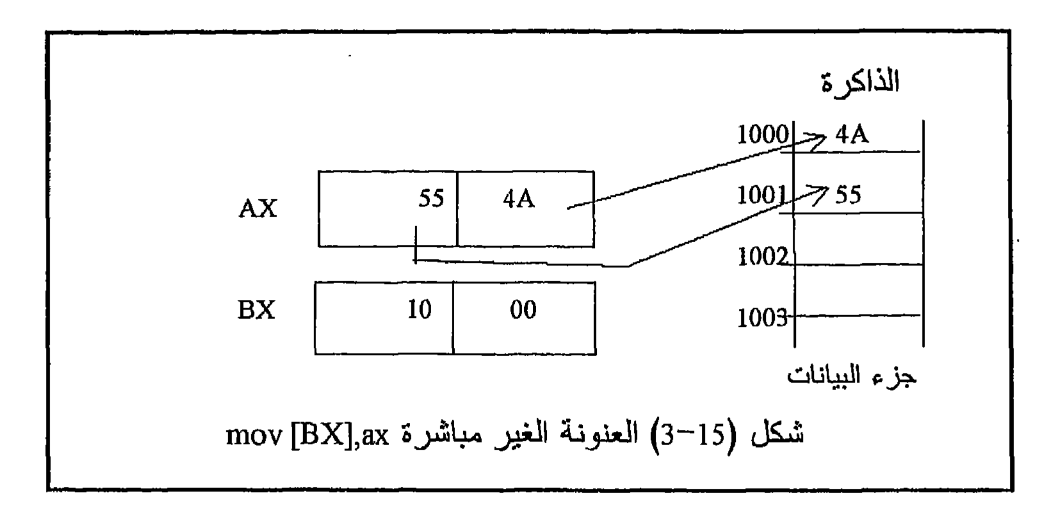
MOV AL,table

والذى يعنى نقل محتويات البايت المسماة بالاسم table إلـــى المســـجل AL . إن جميع العناوين المباشرة تكون دائما محددة فى مقطع أو جزء البيانـــــات . أى أن العنوان 2000H مثلا يحسب ابتداء من العنوان الموجود فى مســـجل التجـــزيء DS.

15-9-2 الطريقة غير المباشرة Indirect addressing

هذه الطريقة من العنونة كما ذكرنا فى الفصل الثاني تسمح بالتعامل مـــع بيانـــات موجودة فى الذاكرة حيث العنوان الذى سيتم التعامل معه فى هذه الحالـــة يكـــون موجودا فى أحد مسجلات المعالج التالية : BX, BP, SI, DI . كمثال على ذلـــك افترض أن المسجل BX يحتوى الرقم 1000H وطلبنا من المعالج تنفيـــذ الأمـــر التالي : MOV AX,[BX] . فى هذه الحالة سيقوم المعالج بإحضار نســـخة مـــن محتويات العنوان 1000H (والذى يليه) الموجود فى المسجل BX ويضعها فـــى المسجل AX . أى أن محتويات المسجل BX الموضوع بين قوسين مربعين كمـــا رأينا تمثل عنوان المعلومة وليس المعلومة نفسها ، ففي عـــدم وجـــود القوســـين

سينسخ المعالج محتويات المسجل BX ويضعها فى المسجل AX كما رأينا فـــى أول طرق العنونة (عنونة المسجل) . يجب أن نؤكد هنـــا أن العنـــوان الفعلـــي للمعلومة يحسب منسوبا لمحتويات مسجل التجزيء DS بعد إزاحته ناحية اليســار 4 بتات كما ذكرنا سالفا ، أى أنه إذا كانت محتويات المسجل DS=0100H فـــإن الأمر السابق سينسخ محتويات العنـــوان 01000+1000=02000H والـــذى يليـــه ويضعها فى المسجل AX . لاحظ أيضـــا أن المســـجلات BX, SI, DI تعنـــون عناوين منسوبة إلى مسجل التجزيء DS بينما المســـجل BP يعنـــون عنـــاوين منسوبة لمسجل التجزيء SS . شكل (3-15) عبارة عن رسم توضيحي لعمليـــة العنونة غير المباشرة للذاكرة . كأمثلة على هذا النوع من العنونة أيضا انظر إلـــى الأوامر التالية :



شكل (3-15) العنونة الغير مباشرة mov [BX],ax

```
MOV CX,[BX]
MOV [BP],BL
MOV [DI],AH
MOV [DI],[BX] (خطأ)
```

حيث الأمر الأول سينقل محتويات عنوان (والذى يليه) فـــى جـــزء البيانـــات أو ذاكرة البيانات data segment والمشار إليه بالمسجل BX إلـــى المســـجل CX ، بينما الأمر الثاني سينقل محتويات النصف الأول من المسجل BX إلـــى عنـــوان مشار إليه بالمسجل BP ويقع فى جزء المكدسة . الأمر الثالث ســـينقل النصـــف العلوي من المسجل AX إلى عنوان مشار إليه بالمسجل DI ويقـــع فـــى جـــزء البيانات ، أما الأمر الثالث فغير مسموح به لأنه ينقل من ذاكرة إلى ذاكرة وهـــذا

النوع من العنونة غير مسموح به إلا فى حالات خاصة جدا مع بعـــض أوامــر سلاسل الحروف .

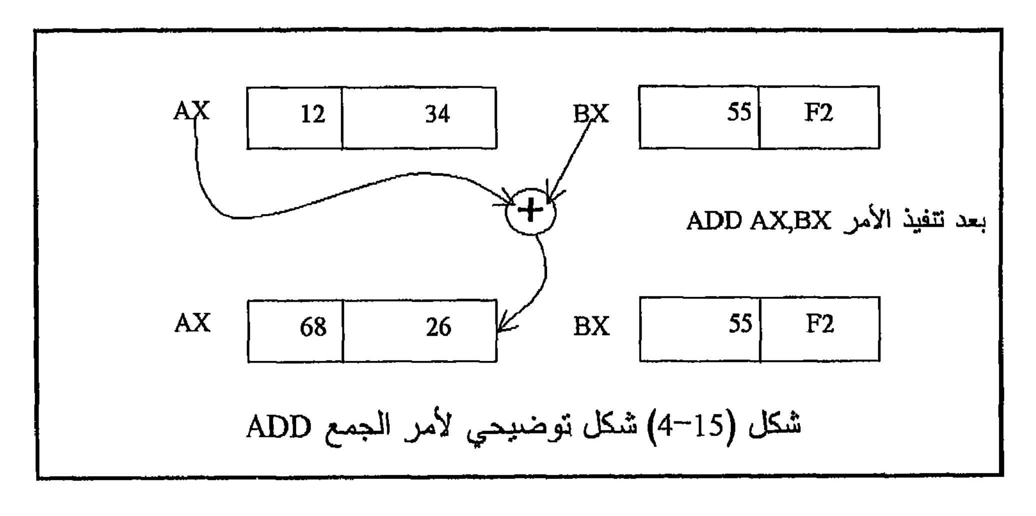
15-10 أوامر الحساب
Arithmetic Instructions

أوامر الحساب التى يمكن تنفيذها بلغة الأسمبلي هى الجمع والطــرح والضــرب والقسمة . كل هذه العمليات يمكن إجراؤها على بيانات 8 بت أو 16بت .

15-10-1 أمر الجمع ADD

الصورة العامة لهذا الأمر هى :

ADD destination, source



شكل (15-4) شكل توضيحي لأمر الجمع ADD

حيث فى هذا الأمر يتم جمع المصدر source مع الهدف destination وتوضـــع النتيجة فى الهدف . من أمثلة ذلك ما يلي :

ADD AX,BX

هذا الأمر يجمع محتويات المسجل BX مع محتويات المسجل AX ويضع النتيجة فى المسجل AX . لاحظ أن هذا الأمر يجمع مسجلين كل منهما 16بت . شـــكل (15-4) يبين رسما توضيحا لهذا الأمر .

أمثلة أخرى على هذا الأمر ما يلي :

ADD Al,CL

الذى يجمع محتويات النصف الأول من المسجل CX مـــع النصـــف الأول مـــن المسجل AL وبضع النتيجة فى المسجل AL .

ADD AL,[BX]

الذى يجمع محتويات مكان الذاكرة (8بت) الذى عنوانه فى المســـجل BX مـــع محتويات المسجل AL ويضع النتيجة فى المسجل AL .

ADD AX,0F437H

هذا الأمر يجمع الثابت أو القيمة الفورية F437H (16بـــت) مـــع المســـجل AX ويضع النتيجة فى المسجل AX . فى العادة يتم وضع 0 قبـــل أى ثـــابت يبـــدأ بحرف وكذلك وضع الحرف H فى نهاية الثابت للدلالة على أنـــه فـــى النظـــام الستعشرى .

ADD table,20H

الذى يجمع الثابت 20H مع محتويات مكان الذاكرة المسمى table ويضع النتيجـة فى نفس المكان . المكان المسمى table تتم تسميته بهذا الاسم فى مقطع البيانـــات فى بداية البرنامج باستخدام أمر التوجيه define . كمثال على ذلك الأمر التالى :

- table DB 22H

  هذا الأمر التوجيهى يحجز بايت اسمها table ويعطيها القيمة الابتدائية 22 .

- table DB 22H, 25H, ‘A’, 33H

هذا الأمر يحجز عدد 5 بايت ويعطيها القيم الابتدائية السابقة ، وهـــذه الخمســة أماكن تأخذ الاسم table . هناك أيضا الأوامر DW الذى يحجـــز كلمـــة word ، والأمر DD الذى يحجز كلمتين ، والأمر DQ الذى يحجز 4 كلمـــات ، والأمـــر DT الذى يحجز 10 كلمات . فى جميع هذه الأوامر الاسم الملحق بـــالأمر يمثـــل عنوان أول بايت فى كمية الذاكرة المحجوزة .

من الأشياء المهمة التي يجب ألا تنسى هي أنه لا يمكن جمع مكان ذاكرة علــــى مكان ذاكرة آخر . فمثلا الأمر التالي خطأ فى عرف لغة الأسمبلى :

ADD [BX],[3F20H]

لأنه يحاول جمع محتويات العنوان 3F20H مع محتويات العنوان الموجـــود فـــى المسجل BX وهذا خطأ أو غير مسموح .

2-10-15 أمر الجمع ADC

الصورة العامة لهذا الأمر هى :

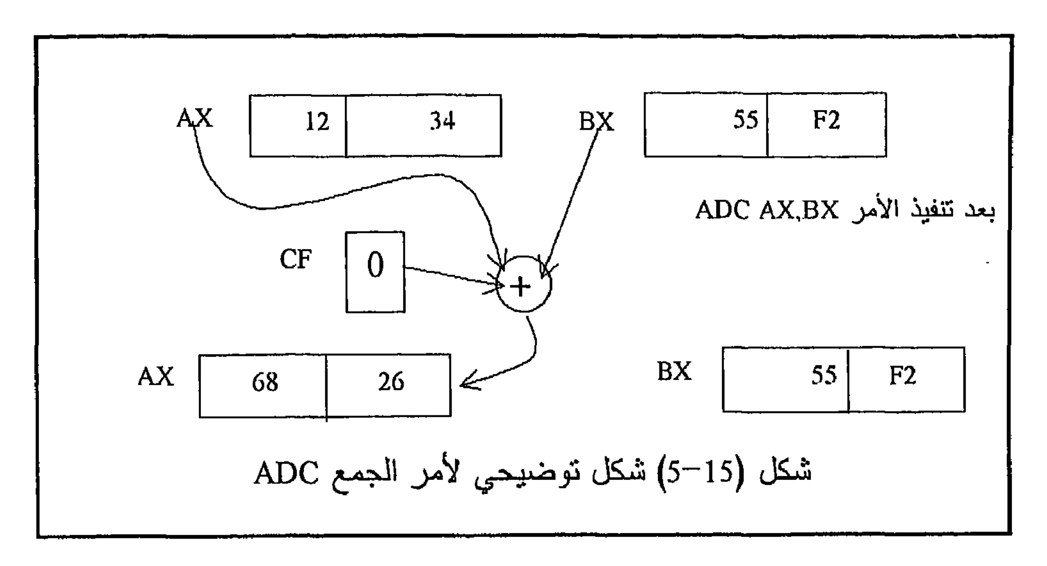
ADC destination, source

حيث فى هذا الأمر يتم جمع المصـــدر source مـــع الـــهدف destination مـــع محتويات علم الحمل carry flag والتي تكون صفرا أو واحد ، وتوضع النتيجـــة فى الهدف . من أمثلة ذلك ما يلي :

ADC AX,BX

هذا الأمر يجمع محتويات المسجل BX مع محتويات المسجل AX مع محتويـــات علم الحمل ويضع النتيجة فى المسجل AX . لاحظ أن هذا الأمر يجمع مســـجلين كل منهما 16بت مع علم الحمل . شكل (5-15) يبين رسما توضيحا لهذا الأمــر . أمثلة أخرى على هذا الأمر ما يلي :

ADC AL,CL
ADC BL, table
ADC CL,40H
ADC BX, [SI]



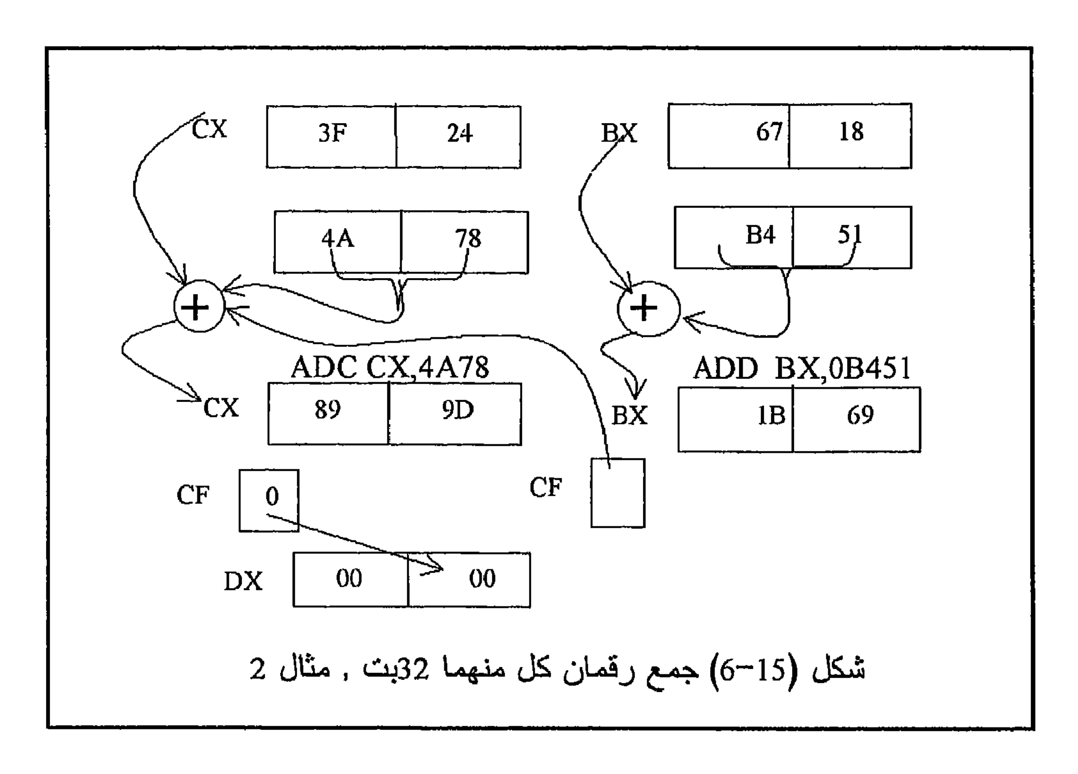
شكل (5-15) شكل توضيحي لأمر الجمع ADC

كل هذه الأوامر تجمع محتويات المصدر مع الهدف مع علم الحمل وتضع النتيجة فى الهدف . هنا يظهر سؤال مهم عن الفرق بين الأمرين ADD و ADC . لكــي نفهم الفرق بينهما نسوق المثال التالي :

مثال 15-4

اكتب برنامجا يجمع الرقمين 3F246718 و 4A78B451 ويضــــع النتيجـــة فـــى المسجلات BX و CX و DX . نلاحظ أن كل من الرقمين مكون من 32بــــت ، وليس هناك وسيلة لجمع رقمين كل منهما 32بت مرة واحدة . لذلك سنضع الرقــم الأول فى المسجلين BX وCX ثم نجمع النصف الأول من الرقم الثاني (B451) مع المسجل BX باستخدام الأمر ADD وبعد ذلك نجمع النصف الثاني من الرقــم

الثاني (4A78) مع المسجل CX باستخدام الأمر ADC حتى يتم أخذ علم الحمل فى الاعتبار لأنه قد يكون هناك حمل من عملية الجمع الأولى فيجب أخذه فى الاعتبار . شكل(15-6) يبين رسما توضيحيا لهذا المثال . لاحظ أن عملية الجمع الأولى يجب أن تتم باستخدام الأمر ADD وليس باستخدام الأمر ADC لأنه لو استخدم الأمر ADC فسيجمع علم الحمل من عملية سابقة مما سيؤثر على النتيجة ويجعلها خطأ .



شكل (15-6) جمع رقمان كل منهما 32بت , مثال 2

البرنامج الذى سيقوم بعملية الجمع السابقة من الممكن أن يكون كالتالي :

```
ADD BX,0B451H
ADC CX,4A78H
MOV DX,00H
ADC DX,DX
```

15-10-3 أمر الطرح SUB

هناك أمران للطرح مثل الجمع ، أحدهما لا يأخذ علم الحمل فى الاعتبار والآخر يأخذ علم الحمل فى الاعتبار . الأمر SUB يطرح محتويات المصدر من محتويات الهدف ويضع النتيجة في الهدف ، فمثلا الأمر :

SUB AX,BX

سيطرح المسجل BX (المصدر) من المسجل AX (الهدف) ويضع النتيجة في المسجل AX . نؤكد هنا على أن الهدف يكون هو دائما المطروح منه .
أمثلة أخرى على أوامر الطرح كالتالي :

```
SUB DL,CL     ;     DL-CL ———> DL
SUB AL,[BX]   ;     AL-[BX] ———> AL
SUB BL,20H    ;     BL-20 ———> BL
```

15-10-4 أمر الطرح SBB

هنا يتم طرح المصدر وعلم الحمل CF من الهدف وتوضع النتيجة في الهدف .
من أمثلة ذلك ما يلي :

```
SBB AX,BX   ;     AX-BX-CF ———>AX
SBB DL,CL     ;     DL-CL-CF ———>DL
SBB AL,[BX]   ;     AL-[BX]-CF ———>AL
SBB BL,20H    ;     BL-20-CF ———>BL
```

يجب أن نكون حذرين جدا فى الأماكن التى نستخدم فيها الأمر SUB والأماكن الأخرى التى يجب أن نستخدم فيها الأمر SBB لأن ذلك من الممكن أن يعطى نتيجة خاطئة كما أوضحنا مع أوامر الجمع .

15-10-5 أمر المقارنة CMP

من الأوامر الشهيرة الاستخدام فى الكثير من التطبيقات أمر المقارنة CMP الذى له الصورة العامة التالية :

CMP destination, source

حيث يتم طرح المصدر source من الهدف destination ولكن هنا لا يتم وضع النتيجة في الهدف ولكن يبقي الهدف دون تغيير ، وهذا هو الاختلاف الأساسي بين هذا الأمر وأوامر الطرح السابقة . الاستفادة الوحيدة من تنفيذ هذا الأمر هى تأثر الأعلام بنتيجة هذا الطرح . بالطبع ما أكثر التطبيقات التى نحتاج فيها لمقارنة رقمين لمعرفة أيهما أكبر من الآخر مثلا دون تغيير قيمة أى واحد من الرقمين حيث فى هذه الحالة فإن أوامر الطرح السابقة لا تؤدي هذا الغرض مباشرة ، لذلك فى هذه الأحوال نستخدم الأمر CMP .

15-10-6 الأوامر INC و DEC

تعمل هذه الأوامر على زيادة محتويات المصدر أو إنقاصها بمقدار واحد . الصورة العامة لهذين الأمرين هى كالتالي :

```
INC source
DEC source
```

ومن أمثلة ذلك ما يلي :

```
INC BL
INC CX
INC [SI]
DEC DL
DEC [BX]
```

وجميعها تجمع 1 أو تطرح 1 من محتويات المصدر الموجود فى الأمر سواء كان مسجل (8 أو 16 بت) أو بايت ذاكرة .

مثال 15-5

أكتب برنامجا يقوم بضرب محتويات المسجلين CL و BL ويضع النتيجة فى المسجل AX وذلك عن طريق الجمع المتكرر .

```
        MOV AL,00
        MOV AL,CL
        DCR BL
YY:     ADD AL,CL
        JNC XX
        INC AH
XX:     DCR BL
        JNZ YY
        HLT
```

مثال 15-6

أكتب برنامجا يقسم محتويات المسجل AL على محتويات المسجل CL ويضع النتيجة فى المسجل AH والباقي فى المسجل DL وذلك عن طريق الطرح المتكرر .

```
        MOV AH,00
        CMP AL,CL
        JS XX ;       CL>AL put AL into DL as a remainder
              ;       and stop
YY:     SUB AL,CL
        JS XX1
        INC AH
        JMP YY
XX1:    ADD AL,CL
XX:     MOV DL,AL  ;  put remainder in DL.
        HLT
```

15-10-7 أمر الضرب MUL

الصورة العامة لهذا الأمر هى :

MUL source

حيث يقوم هذا الأمر بضرب المصدر source الذى يكون مسجلا (8 بت أو 16 بت) أو بايت ذاكرة فى المسجل AL وذلك إذا كان المصدر 8 بت ، أما إذا كان المصدر 16 بت فإنه يضرب المصدر فى المسجل AX . أى أن الطرف الثاني لعملية الضرب يكون دائما إما المسجل AL أو المسجل AX على حسب عدد بتات المصدر . إذا كان المصدر 8 بت فإن الطرف الثاني لعملية الضرب يكون المسجل AL كما ذكرنا وتوضع النتيجة فى المسجلين AL و AH حيث يحتوي AL النصف الأدنى من النتيجة ويحتوى المسجل AH النصف الأعلى منها . إذا كان المصدر 16 بت فإن الطرف الثاني لعملية الضرب يكون المسجل AX كما ذكرنا ، ولكن النتيجة فى هذه المرة توضع فى المسجلين AX (الذى يحتوي الجزء الأدنى من النتيجة) و DX الذى يحتوى الجزء الأعلى منها . بالطبع فإن جميع الأعلام تتأثر بهذه العملية . من أمثلة ذلك ما يلي :

MUL BL

الذى يضرب محتويات المسجل BL فى المسجل AL ويضع النتيجة فى المسجل AX .

MUL CX

الذى يضرب محتويات المسجلين CX و AX ويضع النتيجة فى المسجلين AX و DX .

MUL [BX]

الذى يضرب محتويات بايت الذاكرة المشار إليها بالعنوان الموجود فى المسجل BX فى المسجل AL ويضع النتيجة فى المسجل AX .

من الأشياء المهمة التى يجب أن نحذرها أو نتجنبها هى ضــــرب قيمـــة فوريـــة (ثابت) فى المسجل AL أو AX على الصورة التالية :

MUL 05
MUL 3f24

هذا الشكل لأمر الضرب ممنوع

فى هذه الحالة يجب أن نضع الثابت فى أى مسجل أولا ثم نستخدم هذا المســـجل فى عملية الضرب .

15-10-8 أمر القسمة DIV

الصورة العامة لهذا الأمر هى :

DIV source

حيث يقوم هذا الأمر بقسمة المسجل AX أو المسجلين AX و DX على المصـدر source الذى يكون مسجلا (8 بت أو 16 بت) أو بايت ذاكرة . أى أن المصــــدر يكون دائما هو المقسوم عليه . أما المقسوم فيتحدد على حسب المصدر كالتالي :

1. إذا كان المصدر 8 بت فإن المقسوم لابد أن تكون بتاته ضعف بتات المقســوم عليه ولذلك فإنه يكون المسجل AX فى هذه الحالة . وفى هــذه الحالـــة يخـــزن خارج القسمة فى المسجل AL والباقي من عملية القسمة فى المسجل AH .
2. إذا كان المصدر 16 بت فإن المقسوم يكون المسجلين AX و DX حيـث AX يحتوى النصف الأدنى و DX يحتوى النصف الأعلى للمقسوم . أما ناتج القسمة فإنه يوضع فى المسجل AX والباقي من القسمة فإنه يوضع فى المسجل DX .

من أمثلة عمليات القسمة ما يلي :

DIV BL

الذى يقسم محتويات المسجل AX على المسجل BL ويضع النتيجة فى المســـجل AL والباقي فى المسجل AH .

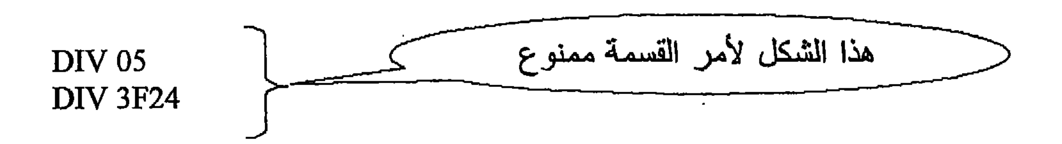
DIV CX

الذى يقسم محتويات المسجلين AX و DX على المسجل CX ويضع النتيجة فـــى المسجل AX والباقي فى المسجل DX .

DIV [BX]

الذى يقسم محتويات المسجل AX على بايت الذاكـــرة المشـــار إليـــها بـــالعنوان الموجود فى المسجل BX ويضع النتيجة فى المسجل AL والباقي فـــى المســـجل AH .

من الأشياء المهمة التى يجب أن نحذرها أو نتجنبها هى استخدام قيمـــــة فوريـــة (ثابت) فى عملية قسمة كالتالي :



فى هذه الحالة يمكن أن يوضع الثابت فى أى مسجل ثم يستخدم هذا المسجل فـــى عملية القسمة .

<u>مثال 15-7</u>

أكتب برنامجا يحسب مضروب الرقم الموجود فى عنوان الذاكرة 1800H ويضع نتيجة هذا المضروب فى المسجل AX .

```
     MOV AX, 01
     MOV BL,[1800]
     MOV BH,00
     MOV CX,BX
     DEC CX
XX:  MUL BL
     DEC BL
     LOOP XX
     HLT
```

15-11 أوامر المنطق
Logic Instructions

هذه المجموعة خاصة بإجراء العمليات المنطقية ، ويقصـــد بالعمليـــة المنطقيـــة العملية التى معاملاتها هى الواحد أو الصفر . العمليات المنطقية التـــى يجريـــها المعالج 8088 هى عمليات AND, OR, XOR, NOT .

<u>15-11-1 الأمر AND</u>

الصورة العامة لهذا الأمر هى :

AND destination, source

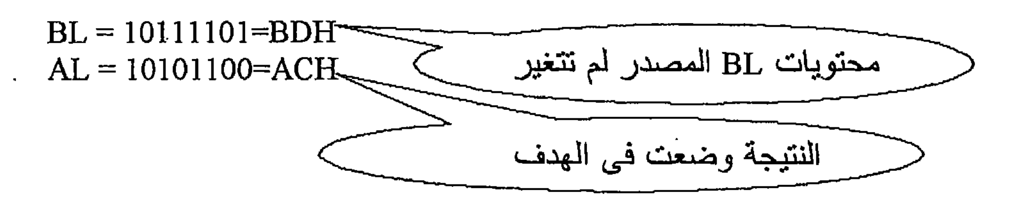
حيث يتم إجراء عملية AND على كل من المصدر source والهدف destination وتوضع النتيجة فى الهدف destination . بالطبع فإن كل من المصدر والـــهدف لابد أن يكونا نفس الحجم أى لهما نفس العدد من البتات . من أمثلة ذلك ما يلي :

AND AL,BL

حيث يجرى عملية AND على محتويات المسجلين AL, BL ويضع النتيجة فـــى المسجل AL . فمثلا افترض أن المسجلين AL, BL يحتويان البيانات التالية :

BL = 10111101=BDH
AL = 11101110=EEH

فإنه بعد تنفيذ الأمر AND AL,BL ستكون محتوياتهما كالتالي :



AND CX,DX
AND BH,table
AND CH,11011011B

حيث الأمر الأخير سيجرى عملية AND على المسجل CH والمعلومة الفورية أو الثابت 11011011B المعطى فى الصورة الثنائية والتى تم الإشارة إليها باســتخدام الحرف B فى آخر الثابت وحيث تسمح لغة التجميع بذلك .

2-11-15 الأمر OR

الصورة العامة لهذا الأمر هى :

OR destination, source

حيث يتم إجراء عملية OR على كل من المصدر source والـــهدف destination وتوضع النتيجة فى الهدف destination . بالطبع فإن كل من المصدر والـــهدف لابد أن يكونا نفس الحجم أى لهما نفس العدد من البتات . من أمثلة ذلك ما يلي :

OR AL,BL

حيث يجرى عملية OR على محتويات المسجلين AL, BL ويضع النتيجـــة فـــى المسجل AL

OR CX,DX

OR BH,table
OR CH,11011011B

15-11-3 الأمر XOR

الصورة العامة لهذا الأمر هى :

XOR destination, source

حيث يتم إجراء عملية XOR على كل من المصدر والهدف وتوضع النتيجة في الهدف . من أمثلة ذلك ما يلي :

XOR AL,BL

حيث يجرى عملية XOR على محتويات المسجلين AL, BL ويضع النتيجة فى المسجل AL .

XOR CX,DX
XOR BH,table
XOR CH,11011011B

15-11-4 أمر النفي المنطقي NOT

هذا الأمر له معامل واحد والصورة العامة له هى كالتالي :

NOT source

حيث يتم عكس المصدر عكسا منطقيا بجعل كل صفر واحد وكل واحد صفر وبالطبع فإن نتيجة هذا العكس توضع في نفس المصدر . من أمثلة ذلك ما يلى :

NOT AL
NOT DX
NOT [BX]

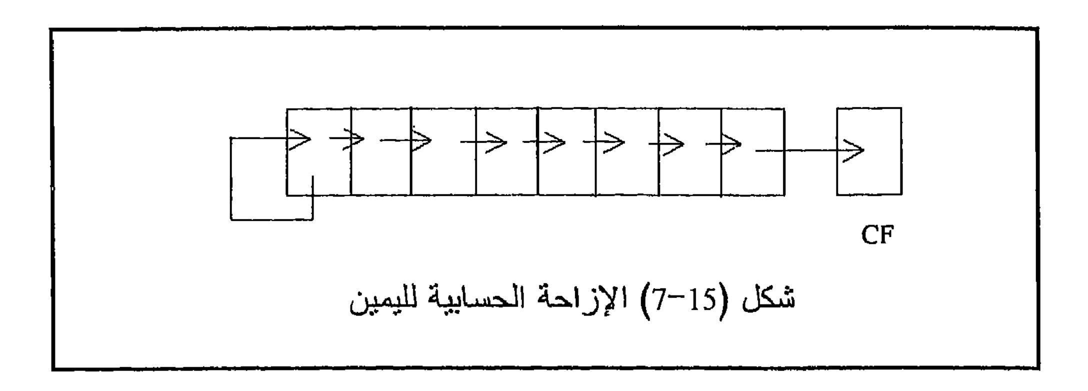
15-12 أوامر الإزاحة والدوران
Shift And Rotate Instructions

هناك نوعان من الإزاحة ، إزاحة حسابية وهي التي يتم الاحتفاظ فيها بإشارة الرقم الذى تجرى عليه الإزاحة وبالذات إذا كانت الإزاحة ناحية اليمين كما سنرى . أما الإزاحة المنطقية فلا اعتبار فيها للإشارة .

15-12-1 أمر الإزاحة الحسابية لليمين SAR

يتم هنا إزاحة محتويات كل خانة إلى الخانة التى على يمينها ، وأما الخانة التى يتم تفريغها فى أقصى اليسار فيتم ملأها دائما بمحتويات خانة الإشارة والتى هى نفس محتويات الخانة الأخيرة . أى أن محتويات آخر بت ناحية اليسار يتم

الاحتفاظ بها دائما ولا تفرغ . أما محتويات البت فى أقصى اليمين فتذهب إلى علم الحمل . شكل (15-7) يبين رسما توضيحيا لهذا الأمر والصورة العامة له ستكون كالتالي :



شكل (15-7) الإزاحة الحسابية لليمين

SAR destination, count

حيث count إما أن تكون واحد إذا كانت الإزاحة ستتم مرة واحدة فقط وأما إذا أردنا الإزاحة أكثر من مرة فيوضع عدد المرات فى المسجل CL ثم يستخدم المسجل CL بدلا من count فى الصورة العامة للأمر . كمثال على ذلك نفترض أن المسجل AL = 10011010 بعد تنفيذ الأمر SAR AL,1 فإن محتويات المسجل AL وعلم الحمل ستكون كالتالي :

AL = 1 0 0 1 1 0 1 0

AL = 1 1 0 0 1 1 0 1 , CF = 0

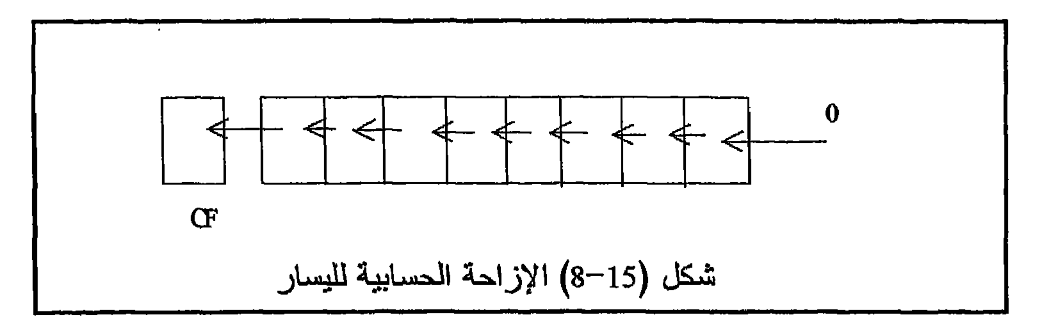
15-12-2 أمر الإزاحة الحسابية لليسار SAL

يتم هنا إزاحة محتويات كل خانة إلى الخانة التى على يسارها ، وأما الخانة التى يتم تفريغها فى أقصى اليمين فيتم ملأها بأصفار . أما محتويات البت فى أقصى اليسار فتذهب إلى علم الحمل . شكل (15-8) يبين رسما توضيحيا لهذا الأمر والصورة العامة له ستكون كالتالي :

SAL destination, count

حيث count إما أن تكون واحد إذا كانت الإزاحة ستتم مرة واحدة فقط وأما إذا أردنا الإزاحة أكثر من مرة فيوضع عدد المرات فى المسجل CL ثم يستخدم المسجل CL بدلا من count فى الصورة العامة للأمر . كمثال على ذلك نفترض

أن المسجل AL= 10011010 بعد تنفيذ الأمر SAL AL,1 فإن محتويات المســـجل AL وعلم الحمل ستكون كالتالي :



شكل (15-8) الإزاحة الحسابية لليسار

AL = 1 0 0 1 1 0 1 0

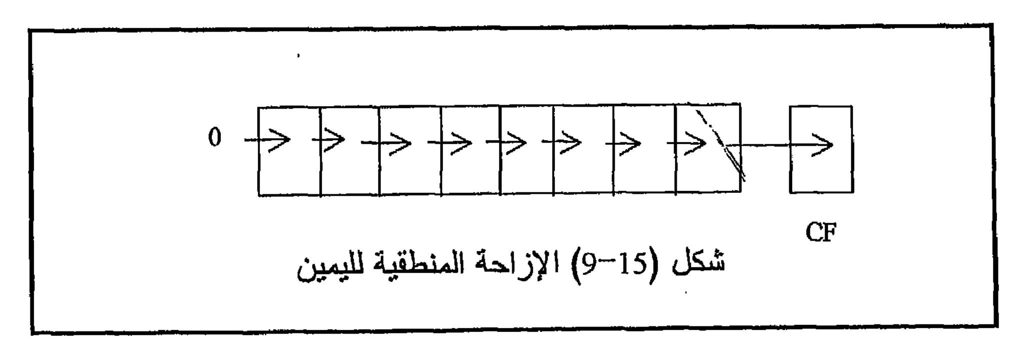
AL = 0 0 1 1 0 1 0 0 <0　　　　CF = 1

15-12-3 أمر الإزاحة المنطقية لليمين SHR

يتم هنا إزاحة محتويات كل خانة إلى الخانة التى على يمينها ، وآخر خانة ناحيــة اليسار يتم ملأها دائما بصفر مع كل إزاحة . أما محتويات البـــت فـــى أقصـــى اليمين فتذهب إلى علم الحمل . شكل (15-9) يبين رسما توضيحيا لـــهذا الأمـــر والصورة العامة له ستكون كالتالي :

SHR destination, count

حيث count إما أن تكون واحد إذا كانت الإزاحة ستتم مرة واحدة فقط وأمـــا إذا أردنا الإزاحة أكثر من مرة فيوضع عدد المرات فى المسجل CL ثـــم يســتخدم المسجل CL بدلا من count في الصورة العامة للأمر . كمثال على ذلك نفترض أن محتويات المسجل هى AL = 10011010 ، بعد تنفيذ الأمر SHR AL,1 فـــإن محتويات المسجل AL وعلم الحمل ستكون كالتالي :



شكل (15-9) الإزاحة المنطقية لليمين

AL = 1 0 0 1 1 0 1 0

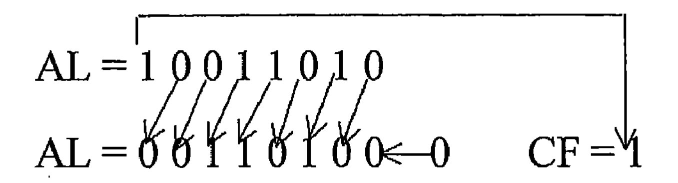
AL = 0 1 0 0 1 1 0 1 , CF = 0

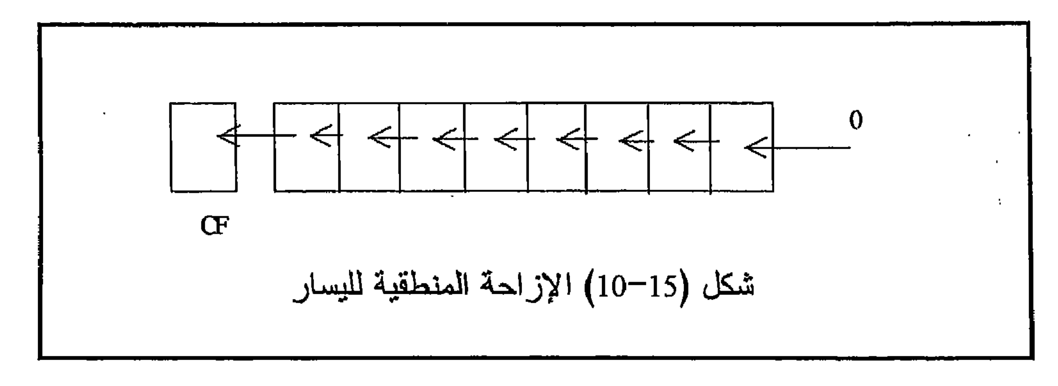
15-12-4 أمر الإزاحة المنطقية لليسار SHL

يتم هنا إزاحة محتويات كل خانة إلى الخانة التى على يسارها ، وأما الخانة التى يتم تفريغها فى أقصى اليمين فيتم ملأها بأصفار . أما محتويات البت فى أقصى اليسار فتذهب إلى علم الحمل تماما مثل الأمر SAL . شكل (15-10) يبين رسما توضيحيا لهذا الأمر والصورة العامة له ستكون كالتالي :

SHL destination, count

حيث count إما أن تكون واحد إذا كانت الإزاحة ستتم مرة واحدة فقط وأما إذا أردنا الإزاحة أكثر من مرة فيوضع عدد المرات فى المسجل CL ثم يستخدم المسجل CL بدلا من count فى الصورة العامة للأمر . كمثال على ذلك نفترض أن المسجل AL = 10011010 بعد تنفيذ الأمر SAL AL,1 فإن محتويات المسجل AL وعلم الحمل ستكون كالتالي :

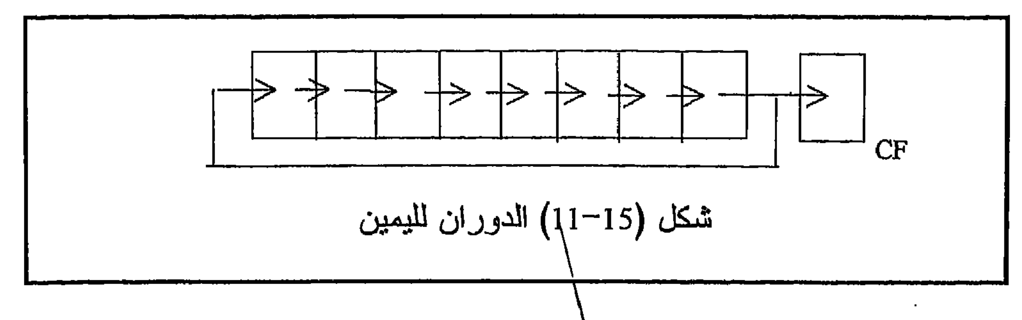




شكل (15-10) الإزاحة المنطقية لليسار

15-12-5 أمر الدوران لليمين ROR

أوامر الدوران تشبه أوامر الإزاحة تماما فيما عدا أن المحتويات التى تزاح مـــن أى جهة يتم إدخالها مرة أخرى من الجهة الأخرى . شكل (15-11) يبين رســـما توضيحيا لأمر الدوران لليمين والصورة العامة له ستكون كالتالي :



شكل (15-11) الدوران لليمين

ROR destination, count

حيث count هى كما شرحنا فى الأوامر السابقة تماما . كمثال على ذلك نفـترض أن المسجل AL = 10011010 بعد تنفيذ الأمر ROR AL,1 فإن محتويات المسجل AL وعلم الحمل ستكون كالتالي :

AL = 1 0 0 1 1 0 1 0

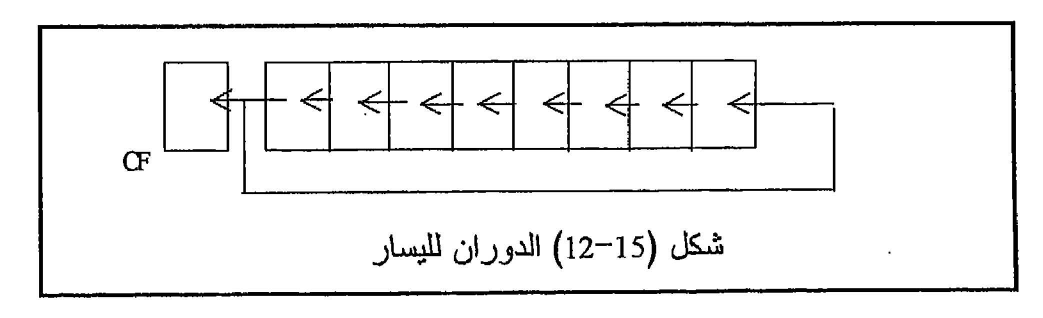
AL = 0 1 0 0 1 1 0 1 ,   CF = 0

15-12-6 أمر الدوران لليسار ROL

شكل (15-12) يبين رسما توضيحيا لهذا الأمر والصورة العامـــة لــه ســتكون كالتالي :

ROL destination, count

كمثال على ذلك نفترض أن المسجل AL = 10011010 حيث بعد تنفيـــذ الأمـــر ROL AL,1 فإن محتويات المسجل AL وعلم الحمل ستكون كالتالي :



شكل (15-12) الدوران لليسار

AL = 1 0 0 1 1 0 1 0

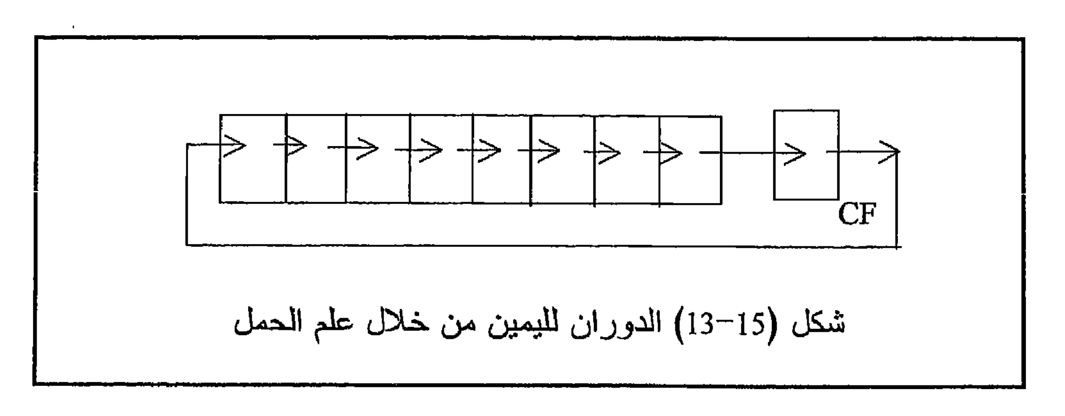
AL = 0 0 1 1 0 1 0 1   CF = 1

15-12-7 أمر الدوران لليمين من خلال علم الحمل RCR

شكل (15-13) يبين رسما توضيحيا لأمر الدوران لليمين من خلال علم الحمـــل والصورة العامة له ستكون كالتالي :

RCR destination, count

حيث count هى كما شرحنا فى الأوامر السابقة تماما . كمثال على ذلك نفــترض أن المسجل AL = 10011010 بعد تنفيذ الأمر RCR AL,1 فإن محتويات المسجل AL وعلم الحمل ستكون كالتالي بافتراض أن علم الحمل كان فيه واحد قبل تنفيـذ الأمر :



شكل (15-13) الدوران لليمين من خلال علم الحمل

AL = 1 0 0 1 1 0 1 0   CF = 1

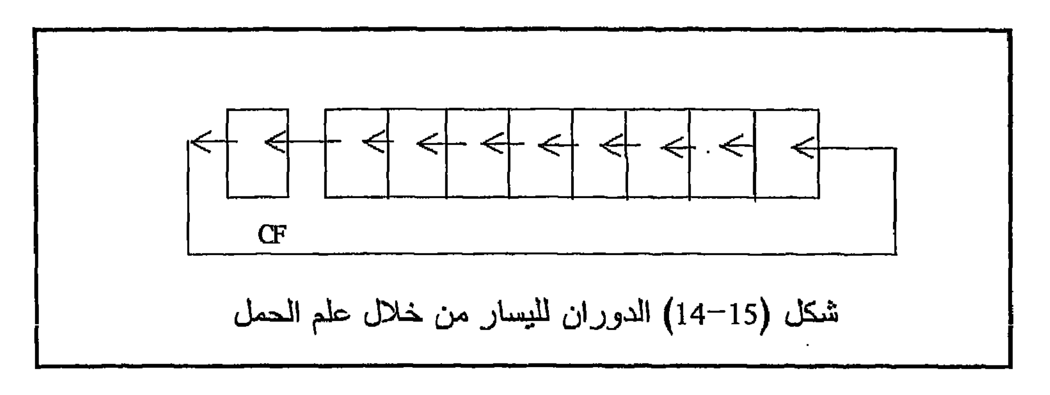
AL = 1 1 0 0 1 1 0 1 ,   CF = 0

15-12-8 أمر الدوران لليسار من خلال علم الحمل RCL

شكل (15-14) يبين رسما توضيحيا لهذا الأمر والصورة العامـــة لـــه ســـتكون كالتالي :

RCL destination, count

كمثال على ذلك نفترض أن المسجل AL = 10011010 حيث بعد تنفيــــذ الأمـــر RCL AL,1 فإن محتويات المسجل AL وعلم الحمل ستكون كالتالي بـلفتراض أن علم الحمل كان واحدا قبل تنفيذ الأمر :



شكل (15-14) الدوران لليسار من خلال علم الحمل

AL = 1 0 0 1 1 0 1 0   CF = 1

AL = 0 0 1 1 0 1 0 1   CF = 1

<u>مثال 15-8</u>

أكتب برنامج يحسب مضروب الأرقام المخزنة فى الذاكـــرة مـــن 1800H إلـــى 1810H ويخزن المضروب فى الأماكن 1900H إلى 1910H . استخدم الـــبرامج الفرعية معتبرا أن قيمة المضروب لن تزيد عن بايت واحدة .

```
        MOV DI,1800
        MOV SI,1900
        MOV CX,10H
        MOV BH,00
XX:     MOV BL,[DI]
        PUSH CX
        CALL FACT
        POP CX
        MOV [SI],AL
        INC SI
        MOV [SI],AH
        INC SI
```

```
        INC DI
        LOOP XX
        HLT
FACT:   MOV AL,01
        MOV CX,BX
        DEC CX
YY:     MUL BL
        DEC BL
        LOOP YY
        RET
```

13-15 تمارين

1. هل يمكن جمع المسجلين CX و DS ؟
2. إذا كان المسجل CX=1001H والمسجل DX=20FFH فما هى محتويات كل من المسجلين وكذلك قيمة كل علم من الأعلام بعد تنفيذ أمر الجمع ADD عليهما؟
3. كرر السؤال السابق فى حالة تنفيذ أمر الطرح SUB ؟
4. كرر السؤال الثانى فى حالة إجراء الأوامر التالية AND و OR و XOR و NOT ؟
5. أكتب برنامج يضع صفرا فى جميع أماكن الذاكرة 2000H إلى 20FFH ، وبعد ذلك يختبر نفس الأماكن ليرى إذا كان الصفر ما زال موجودا أم لا ؟ هل يصلح هذا ليكون بمثابة اختبار لذاكرة الحاسب ؟
6. اكتب برنامج ينسخ محتويات بلوك الذاكرة 2000H إلى 20FFH فى البلوك 3000H إلى 30FFH ؟
7. أكتب برنامجا يختبر محتويات أماكن الذاكرة 1800H إلى 18FFH ويخرج منها الأعداد الفردية ويخزنها فى الذاكرة من 1900H إلى 19FFH والأعداد الزوجية يخزنها فى الذاكرة من 2000H إلى 20FFH .
8. اكتب برنامج يفحص محتويات الذاكرة من 2000H إلى 2500H ويبحث فيها عن عدد مرات تكرار الرقم 33H المكون من بايت واحدة؟
9. اكتب برنامج يفحص محتويات الذاكرة من 2000H إلى 2500H ويبحث فيها عن عدد مرات تكرار الرقم 33FFH المكون من 2 بايت؟
10. أكتب برنامجا يحسب عدد الوحايد فى كل بايت من بايتات الذاكرة من 1800H إلى 18FFH ويخزنها فى الذاكرة من 1900H إلى 19FFH .

11. أعد السؤال السابق مستخدما البرامج الفرعية .

12. أكتب برنامجا يحسب مضروب الأرقام الموجودة فى الذاكرة من 1800H إلى 18FFH ويخزن هذا المضروب فى الذاكرة من 1900H إلى 19FFH . استخدم البرامج الفرعية .

13. أكتب برنامج يحسب عدد الأرقام السالبة وعدد الأرقام الموجبة فى المدى العنوانى من 1800-18FFH .

14. أكتب برنامج يقرأ محتويات المدى العنوانى 1700-17FFH ثم ينقل البيانات السالبة منها ويخزنها ابتداء من العنوان 1800H والبيانات الموجبة يخزنها ابتداء من العنوان 1900H .

15. أكتب برنامج يحسب عدد الأرقام الفردية وعدد الأرقام الزوجية فى المدى العنوانى 1700-17FFH .

16. اكتب برنامج يبحث عن أكبر رقم من بايت واحدة فى المدى العنوانى 2000H إلى 20FFH ؟

17. اكتب برنامج يقوم بترتيب الأرقام (كل رقم بايت واحدة) الموجودة فى المدى العنوانى 2000H إلى 20FFH ترتيبا تصاعديا؟

18. المدى العنوانى 2000H إلى 3000H يحتوى بيانات إشارة صوتية ، والمطلوب حساب عدد مرات عبور هذه الإشارة للقيمة صفر؟

19. استخدم البرامج الفرعية فى حساب المعادلة التالية :

$$M=20! + 30! + 40! + 24^3 + 25^5 + 10^6$$

الفصل السادس عشر

مواجهة المعالج

8086/8088

16-1 مقدمة

لقد رأينا في الفصول السابقة كيفية مواجهة الشريحة 8085 مع الذاكرة ومع بوابات الإدخال و الإخراج . إن مواجهة الشريحة 8086 لن تختلف كثيرا عن ذلك وسنحاول تقديمها هنا سريعا من خلال شرح مختصر لوظيفة كل طرف من أطراف هذه الشريحة . ثم شرح عملية فصل المسارات المختلفة ، مسار العناوين ومسار البيانات ومسار التحكم .

16-2 الوظائف المختلفة لأطراف الشريحة 8086/8088

كل من الشريحتين 8086 أو 8088 لها 40 طرفا وهذه هي آخر أجيال شرائح المعالجات التي لها هذا العدد من الأطرف حيث كما سنرى في المعالجات القادمة أن عدد الأطراف قد قفز قفزة كبيرة . الاختلاف بين وظائف أطراف الشريحة 8086 و الشريحة 8088 بسيطا جدا وفي عدد محدود جدا من الأطراف كما سنرى .

قبل أن ندخل في تفاصيل وظائف هذه الأطراف يجب أن نعلم أن كل من الشريحتين تستخدم فكرة المزج الزمني time multiplexing التي سبق شرحها مع الشريحة 8085 بإسهاب ومع الكثير من الأطراف . إن فكرة المزج الزمني كما سبق وقدمناها تتلخص في أن نفس الطرف يمكن أن يحمل أكثر من إشارة في وقتين مختلفين . فمثلا الطرف AD0 يحمل إشارة عناوين وإشارة بيانات أيضا ، فهذا الخط يمثل الإشارة D0 عند لحظة معينة ، كما يمثل الإشارة A0 عند لحظة أخرى . قبل مواجهة شريحة المعالج مع أية شريحة أخرى لابد من فصل إشارة البيانات على خط منفصل وإشارة العناوين على خط آخر . تتم عملية الفصل باستخدام الطرف ALE الذي يكون واحدا عندما تكون الإشارة على الطرف AD0 مثلا تمثل عناوين ، ويكون صفرا عندما تكون الإشارة على هذا الخط تمثل بيانات .

هنا نرجو من القارئ أن يراجع الفصل الخاص بفصل أو عزل مسارات الشريحة 8085 حيث أنها هي نفسها بالضبط المستخدمة مع الشريحة 8086/8088 .

شكل (16-1) يبين رسما طرفيا لكل من الشريحتين 8086/8088 حيث يمكن عرض وظائف هذه الأطراف كما يلي :

1. الخطوط AD0 إلى AD7 تحمل مزيج من إشارة العناوين و البيانات حيث تكون الإشارة على هذه الأطراف عناوين عندما يكون الطرف ALE فعال أي واحد ، وتكون بيانات عندما يكون الطرف ALE غير فعال أي صفر. هذا الكلام مطبق على كل من الشريحتين 8086 أو 8088 .

أ- الشريحة 8086

Signal	Pin	Pin	Signal
GND	1	40	Vcc
AD14	2	39	AD15
AD13	3	38	A16/S3
AD12	4	37	A17/S4
AD11	5	36	A18/S5
AD10	6	35	A19/S6
AD9	7	34	$\overline{BHE}$/S7
AD8	8	33	MN/$\overline{MX}$
AD7	9	32	$\overline{RD}$
AD6	10	31	HOLD
AD5	11	30	HOLDA
AD4	12	29	$\overline{WR}$
AD3	13	28	M/$\overline{IO}$
AD2	14	27	DT/$\overline{R}$
AD1	15	26	$\overline{DEN}$
AD0	16	25	ALE
NMI	17	24	$\overline{INTA}$
INTR	18	23	$\overline{TEST}$
CLK	19	22	READY
GND	20	21	RESET

ب- الشريحة 8088

Signal	Pin	Pin	Signal
GND	1	40	Vcc
A14	2	39	A15
A13	3	38	A16/S3
A12	4	37	A17/S4
A11	5	36	A18/S5
A10	6	35	A19/S6
A9	7	34	$\overline{SS0}$
A8	8	33	MN/$\overline{MX}$
AD7	9	32	$\overline{RD}$
AD6	10	31	HOLD
AD5	11	30	HLDA
AD4	12	29	$\overline{WR}$
AD3	13	28	IO/$\overline{M}$
AD2	14	27	DT/$\overline{R}$
AD1	15	26	$\overline{DEN}$
AD0	16	25	ALE
NMI	17	24	$\overline{INTA}$
INTR	18	23	$\overline{TEST}$
CLK	19	22	READY
GND	20	21	RESET

شكل (16-1) الرسم الطرفي للمعالجين 8086/8088 في الحالة الحقيقية أو الصغرى Minimum mode .

2. الخطوط AD8 إلى AD15 ، في حالة الشريحة 8088 تسمى هذه الأطراف A8-A15 حيث أنها في هذه الحالة تحمل إشارة عناوين فقط طول الوقت وليس هناك أي مزج زمني في الإشارات لأن مسار البيانات في هذه الشريحة 8 خطوط فقط وينتهي عند الطرف AD7 . أما في حالة الشريحة 8086 فإن مسار البيانات يكون 16 طرفا ، لذلك فإن الخطوط AD8-AD15 تحمل مزيجا من إشارة البيانات D8 إلى D15 وإشارة العناوين A8 إلى A15 حيث تكون الإشارة على هذه الخطوط إشارة عناوين عندما يكون الطرف ALE فعالا (1) ، وتكون

الإشارة بيانات عندما يكون الطرف ALE غير فعال (0) كما في حالة الخطـــوط AD0-AD7 .

3. الأطراف A16/S3 و A17/S4 و A18/S5 و A19/S6 تحمل إشارة عنـــاوين للخطوط A16-A19 حينما يكون الخط ALE فعالا . عندما يكـــون الخـــط ALE غير فعال فإن هذه الخطوط تحمل الإشارات S3, S4, S5, S6 التي تمثل حـــالات مختلفة للمعالج . حيث الطرف S6 يكون صفرا دائما ، والطرف S5 يبين حالـــة علم المقاطعة ، والطرفين S3, S4 يبينان أي مقطع من الذاكرة يتم التعامل معـــه في حالة التعامل مع الذاكرة كما في جدول 16-1 .

S4	S3	الوظيـــــفة
0	0	Extra segment
0	1	Stack segment
1	0	Code
1	1	Data

جدول 16-1

4. الطرف $\overline{RD}$ هذا الطرف يكون فعالا (0) حينما يكون المعالج في حالة قـــراءة للبيانات سواء من الذاكرة أو من بوابة إدخال . أي أن البيانات الموجـــودة علـــى مسار البيانات تكون داخلة للمعالج ولا يهم من أي مصدر تكون هذه البيانات .

5. الطرف READY . لكي يقوم المعالج بتنفيذ أي أمر ، لابد وأن يكـــون هـــذا الطرف فعالا (1) . إذا أصبح هذا الخط صفرا فإن المعـــالج يدخـــل فـــي حالـــة انتظار، حيث تتجمد جميع أطراف المعالج على الحالة التي كان عليها ، ويســـتفاد بذلك عند مواجهة المعالج مع بعض الأجهزة الخارجية البطيئة مثل بعض شـــرائح الذاكرة البطيئة أو أجهزة الإدخال والإخراج مثل الطابعـــات والتـــي لا تضـــاهي سرعتها سرعة المعالج .

6. الطرف INTR ، حينما يكون هذا الطرف فعالا (1) تتم مقاطعـــة المعـــالج ، حيث يكمل تنفيذ الأمر المشغول به ثم يذهب لتنفيذ برنامج معيـــن لخدمـــة هـــذه المقاطعة ، وبعد الانتهاء من هذه الخدمة يرجع المعالج إلي نفس مكان البرنـــامج الأساسي الذي تمت عنده المقاطعة .

7. الطرف $\overline{TEST}$ ، الأمر WAIT والذي يمثل حالة انتظار يدخل المعالج فيـــها ويظل فيها طالما أن الخط TEST غير فعال (1) . عندما يصبح هذا الخط فعـــالا (0) فإن المعالج يخرج من حالة الانتظار ويستمر في تنفيذ البرنامج الأساســـي . حالة الانتظار هنا يتم الدخول فيها بالأمر WAIT على العكس من حالة الانتظـــار

التي يتم الدخول فيها بجعل الخط READY صفرا ، حيث تنتـــهي هـــذه الحالـــة برجوع الخط READY واحد مرة أخرى .

8. الطرف NMI ، وهو طرف المقاطعة الغير قابلة للحجب أو غـــير المقنعـــة ، حيث تتم مقاطعة المعالج هنا بانتقال الخط NMI من الصفر إلي الواحد (أي عنـــد الحافة الصاعدة) ، ولايهم هنا أن يكون علم المقاطعة يساوي واحـــد ، وتوضـــع على هذا الطرف إشارات المقاطعة المهمة مثل إشارات انقطاع القدرة مثلا .

9. الطرف RESET ، أو إعادة الوضع ، حيث عندما يكون هذا الطـــرف واحـــد فإن المعالج يذهب فورا إلي عنوان الذاكرة FFF0H ويبدأ التنفيذ من هناك . وفـــي العادة يكون عند هذا العنوان أمر قفز إلي مكان آخر في الذاكرة يحتـــوي شـــفرة برنامج خاص بإعادة الوضع للمعالج .

10. الطرف CLK يتم إدخال نبضات التزامن من علـــى هـــذا الطـــرف بـــالتردد المطلوب و duty cycle أو زمن ON/OFF بنسبة 33% حتى نضمن الـــتزامن المطلوب لجميع العمليات التي ينفذها المعالج .

11. الطرف Vcc حيث يوضع جهد القدرة الثابت 10% -/+ 5V .

12. الطرف GND وهما طرفان يجب أن يكونا متصلين بباقي أطراف الأرضـــى في النظام الخارجي .

13. $MN/\overline{MX}$ ، وهو طرف الحالة حيث عندما يكون هذا الطرف (1) فإن المعالج يعمل في الحالة الصغرى Minimum ، وإذا كان هذا الطرف (0) فإن المعـــالج يعمل في الحالة العظمى Maximum . الحالة الصغرى هي الحالة العادية للمعالج 8086 ، وأما الحالة العظمى فهي حالة المعالج عندما يتعامل مع معالج العمليـــات الحسابية 8087 .

14. الطرف $\overline{BHE}/S7$ يستخدم هذا الخط لتنشيط البايت ذات القيمة العظمـــى مـــن مسار البيانات D0-D15 في أثناء قراءة أو كتابة البيانات . هذه الإشارة تمـــتزج مع إشارة الحالة S7 . هذا الطرف بالطبع موجود فقط في المعالج 8086 الـــذي يتعامل على أساس أن البيانات الخارجية 16 بت .

15. الطرف $IO/\overline{M}$ فى المعالج 8088 هذا الطرف يبين مـــا إذا كـــانت الإشـــارة الموجودة على مسار العناوين تمثل عنوان في ذاكرة أم عنوان لوحدة إدخـــال أو إخراج . هذا الخط يكون (1) إذا كانت الإشارة تمثل عنوان في وحدة إدخـــال أو إخراج و يكون (0) إذا كانت الإشارة تمثل عنوان في ذاكـــرة . لاحـــظ وجـــود شرطة على الحرف M في اسم هذا الطرف . فى حالـــة المعـــالج 8086 تكـــون الإشارة على هذا الطرف معكوسة ورمز الخط فى هذه الحالة هو M/IO .

16. الطرف $\overline{WR}$ مثل الطرف $\overline{RD}$ يكون فعالا (0) في حالة كتابـــة بيانـــات فـــي الذاكرة أو أي وحدة إخراج .

17. الطرف $\overline{INTA}$ ، عند الاستجابة لطلب المقاطعة على الطرف INTR ، فإن المعالج يجعل الخط INTA فعالا (0) للدلالة على أنه قبل المقاطعة واعترف بها وينتظر إدخال رقم المقاطعة التي سيقوم بخدمتها على مسار البيانات .

18. الطرف ALE ، وهو خط مسك العنوان Address latch enable يستخدم للدلالة على أن الإشارة الموجودة على الخطوط AD0-AD15 تمثل عناوين أم بيانات . عندما يكون هذا الخط واحد فإن الإشارة على هذه الخطوط تمثل عنوان وعندما يكون الخط ALE صفرا فإن الإشارة على هذه الخطوط تمثل بيانات . يستخدم هذا الخط لفصل إشارة العناوين عن البيانات كما سنرى .

19. الخط $DT/\overline{R}$ ، خط إرسال أو استقبال البيانات حيث يستخدم هذا الخط لبيان إذا كانت البيانات خارجة Transmit, T أو داخلة Receive, R للمعالج . لذلك فإن هذا الخط سيستخدم لتحديد اتجاه البيانات عند فصل مسار البيانات كما سنرى.

20. الطرف $\overline{DEN}$ ، طرف تنشيط مسار البيانات Data bus enable ، حيث يستخدم هذا الطرف لبيان إذا كانت الخطوط AD0-AD15 تحمل إشارة بيانات حقيقية أم لا .

21. الطرف HOLD ، عندما يكون هذا الطرف فعالا (1) فإن المعالج يضع كل مساراته (البيانات ، والعناوين ، والتحكم) في الحالة الثالثة ، حالة المقاومة العالية ، أو بمعنى آخر فإن المعالج ينفصل عن كل المسارات الخارجية تاركا إياها في العادة لمتحكم الاتصال المباشر بالذاكرة لاستخدامها في نقل البيانات مباشرة للذاكرة .

22. الطرف HLDA ، خط استجابة للطرف HOLD ويستخدمه المعالج للدلالة على أنه قبل الإشارة HOLD وأنه قد تم الانفصال عن جميع المسارات حيث عند ذلك يقوم المعالج بوضع (1) على هذا الخط .

23. الطرف $\overline{SS0}$ في الشريحة 8088 عبارة عن طرف حالة حيث يستخدم مع الأطراف $IO/\overline{M}$ و $DT/\overline{R}$ لبيان حالة المعالج في أثناء أي دورة مسار Bus cycle . جدول 16-2 يبين هذه الحالات .

24. الأطراف S0, S1, S2 عبارة عن أطراف حالة تبين حالة المعالج في أثناء أي دورة مسارات وتستخدم هذه الخطوط في الحالة العظمى للمعالج فقط Maximum mode . جدول (16-3) يبين هذه الحالات ، لاحظ التشابه الموجود بينها وبين جدول 16-2 .

25. الأطراف $\overline{RQ}/\overline{GT1}$, $\overline{RQ}/\overline{GT0}$ تستخدم هذه الخطوط لطلب المسارات عن طريق المعالج الحسابي المساعد Math coprocessor الخارجي في الحالة العظمى فقط maximum mode . كل من هذه الخطوط ثنائي الاتجاه حيث عليها يطلب المعالج المساعد المسارات ، وعليها أيضا يستجيب المعالج ليخبر المعالج

الخارجي بأن المسارات متاحة . هذه الخطوط ممزوجة مع الإشـارات HOLD و HOLDA .

الوظيفة	$\overline{SS0}$	$DT/\overline{R}$	$IO/\overline{M}$
حالـة اعـتراف بالمقاطعـة Interrupt acknowledge	0	0	0
قراءة من الذاكرة Memory read	1	0	0
كتابة فى الذاكرة Memory write	0	1	0
توقف Halt	1	1	0
تعامل مع شفرة أمر Code access	0	0	1
قراءة من بوابة إدخال I/O read	1	0	1
كتابة فى بوابة إخراج I/O write	0	1	1
حالة خمول Remains passive	1	1	1

جدول 16-2

الوظيـــفة	$\overline{S0}$	$\overline{S1}$	$\overline{S2}$
حالـة اعـتراف بالمقاطعـة Interrupt acknowledge	0	0	0
قراءة من بوابة إدخال I/O read	1	0	0
كتابة فى بوابة إخراج I/O write	0	1	0
توقف Halt	1	1	0
تعامل مع شفرة أمر Code access	0	0	1
قراءة من الذاكرة Memory read	1	0	1
كتابة فى الذاكرة Memory write	0	1	1
حالة خمول Remains passive	1	1	1

جدول 16-3

26. الطرف $\overline{LOCK}$ ، يستخدم فى الحالة العظمى فقط (طرف 29) ، حينما يكـون هذا الطرف فعالا (0) يمنع أي معالج خارجي من الاتصال بالمسارات . بعـــض الأوامر تستفيد بهذا الخط حتى تضمـــن اكتمـــال تنفيذهـــا دون أي تدخـــل مـــن المعالجات الخارجية .

27. الأطراف QS0-QS1 (الأطراف 25و24) عبارة عن خطوط حالة تبين حالـــة الطابور Queue الموجود في وحدة مواجهة المسارات لبيان إذا كان هذا الطـابور فارغ أو ممتلئ . لاحظ أن هذا الطابور يتكون من 6 بايت فـــي حالـــة المعـالج 8086 و 4 بايت في حالة المعالج 8088 . هذه الخطوط فعالة فى الحالة العظمـى فقط

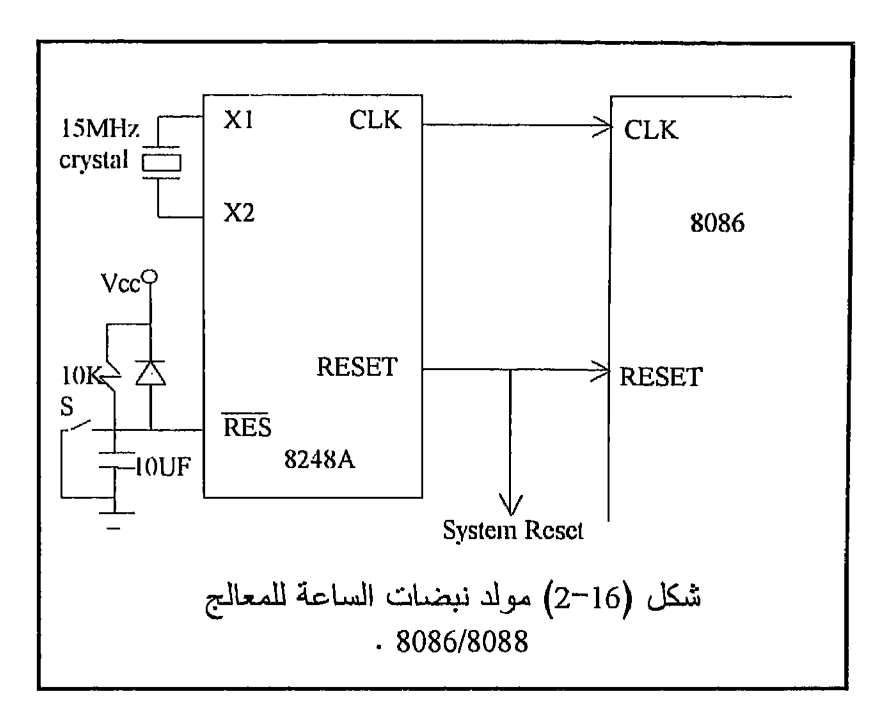
16-2-1 نبضات الساعة clock للمعالج 8086

كما رأينا سابقا فإنه لابد من إدخال نبضات ساعة مربعة الشكل ســابقة التجـهيز إلى الطرف CLK وهو الطرف رقم 19 في الشريحة . تـــردد هـــذه النبضـــات حوالي 5 ميجاهرتز ، و النسبة الدورية لها 33% . هذه النبضـــات يتـم تزامـن جميع عمليات المعالج معها . لقد تم إنتاج شريحة بواسطة شـــركة Intel تعطـــى هذه النبضات بالمواصفات المطلوبة للمعالج 8086 كما أنها تأخذ الإشـــارة Reset من المستخدم وتقدمها للمعالج بالمواصفات و التزامن المطلوب ، وكذلك الإشـــارة READY . هذه الشريحة هى الشريحة Intel 8284A . هذه الشريحة لها طرفـان X1 و X2 يوصل عليها بللورة crystal بالتردد المناسب ، كما يوصـــل عليـــها المفتاح RESET كما فى شكل (16-2) . بالطبع ليس بـــالضرورة اســـتخدام الشريحة 8284A بالذات ولكن يمكن استخدام أى مولـــد ذبذبـــات بالمواصفـــات المطلوبة ، ولكن يفضل استخدامها بالذات تجنبا للكثير من المشاكل وتوفير الكثـير من المكونات التى قد تكون مضطرا لاستخدامها .

16-3 عزل مسارات المعالج 8086

إن عملية عزل (فصل) مسارات المعالج 8086 هى نفسها تمامـــا عمليـــة عـــزل مسارات المعالج 8085 حيث أن كل من المعالجين يستخدم فكرة المزج الزمنـــى لمسارى البيانات والعناوين . لذلك فإننا نحيل القارئ هنا لمراجعة عملية فصـــل كل من مسار العناوين والبيانات للشريحة 8085 والتى سبق شرحها . شكل (16-3) يبين كيفية فصل مسارى العناوين والبيانات وبعض خطوط التحكم للشـــريحة 8086 . نلاحظ من هذا الشكل استخدام ثـــلاث شـــرائح 74373 لمســـك إشـــارة العناوين فى الفترة التى يكون فيها الطرف ALE فعالا (1) . لاحظ اتصال هـــذا الطرف بالثلاث شرائح ، أما الطرف OE لكل شريحة فتم توصيـــله بـــالأرضى حتى يكون خرج هذه الشرائح فعالا دائما . بذلك نضمن مســـك إشـــارة خطـــوط العناوين A19 - A0 وفصلها عن إشارة البيانات . نلاحظ أيضا فى نفس الشـــكل

استخدام شريحتين 74245 للحصول على إشارة البيانات . الشريحة 74245 كما نعلم ثنائية الاتجاه ولذلك فإن اتجاه البيانات فيها تم التحكم فيه باستخدام الطرف DT/$\overline{R}$ الذى تم توصيله على خط التحكم فى الاتجاه DIR فى الشريحة 74245 بذلك نضمن مرور البيانات فى الاتجاه السليم حسب خروجها من المعالج . لابد أيضا أن تكون الشريحة 74245 فعالة فقط أثناء وجود إشارة بيانات محققة على الأطراف AD15-AD0 لذلك تم استخدام الطرف $\overline{DEN}$ الذى يكون فعالا (0) كما ذكرنا من قبل عند وجود بيانات محققة على هذه الأطراف . لذلك تم توصيل الطرف $\overline{DEN}$ بالطرف CS للشريحة 74245 .



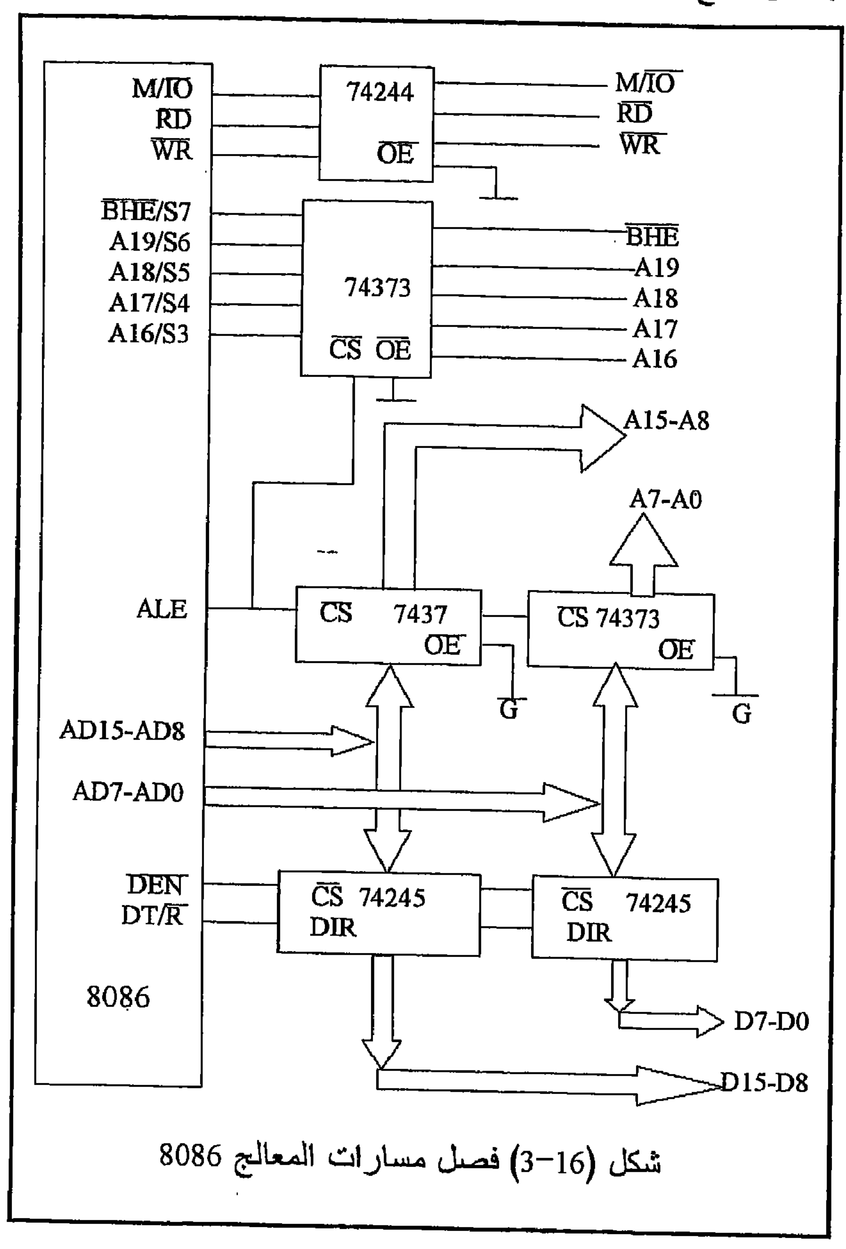
شكل (2-16) مولد نبضات الساعة للمعالج 8086/8088 .

لقد تم استخدام الشريحة 74244 لفصل بعض خطوط التحكم مثل الخطوط IO/$\overline{M}$, $\overline{RD}$, $\overline{WR}$ وهذه الخطوط غير ممزوجة زمنيا مع أى إشارات أخرى لذلك فعملية الفصل هنا تكون من قبيل عمل حساب الأحمال التى ستوصل على هذه الخطوط .

4-16 مواجهة الشريحة 8086/8088 مع الذاكرة

المعالج 8088 له مسار بيانات 8 بت ومسار عناوين 20 بت ، وعلى ذلك فإنه يستطيع التعامل مع ذاكرة مقدارها 1 ميجابايت . و تتم مواجهته مع هذه الذاكرة

بنفس الطريقة التي درسناها مع أي معالج من المعالجات 8 بت التي سبق دراستها ، وننصح هنا بمراجعة الفصل الخاص بمواجهة الذاكرة .



شكل (3-16) فصل مسارات المعالج 8086

كما رأينا فإن شرائح الذاكرة القابلة للقراءة فقط ROM يكون لها خطي تحكم في العادة وهما الخط $\overline{CS}$ أو أحيانا يسمى $\overline{CE}$ وهو خط اختيار الشريحة Chip Select أو خط تنشيط الشريحة Chip Enable حيث لابد أن يكون هذا الطرف

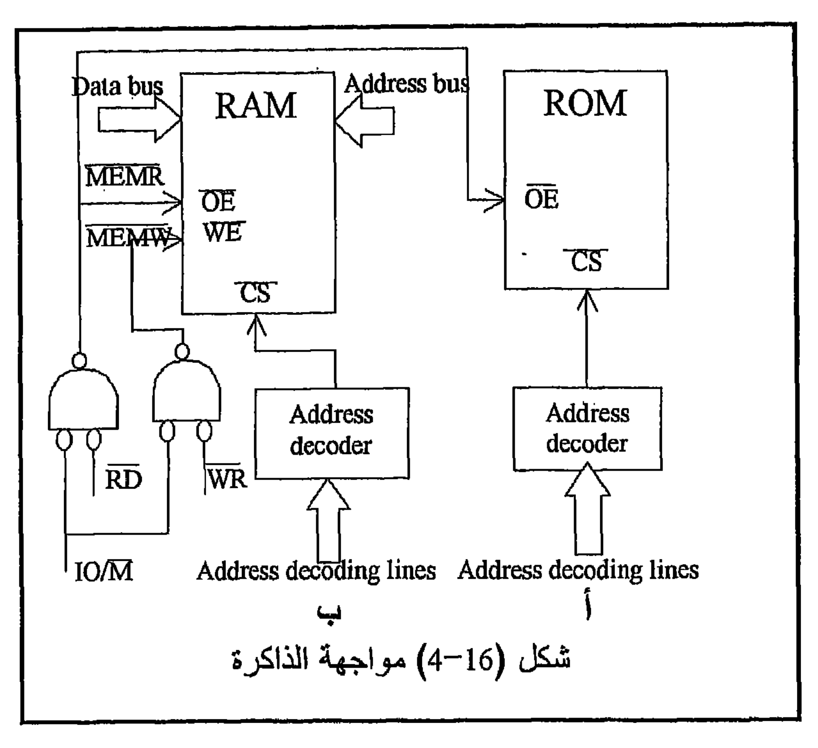
فعالا (0) حتى تعمل هذه الشريحة . هذا الطرف يوصل في العادة بخرج مشفر العناوين Address Decoder الذي نضمن به أن هذه الشريحة لن تعمل إلا في حالة وجود عنوان يقع ضمن المدى العنواني لهذه الشريحة والمحدد بمشفر العناوين . خط التحكم الثاني في شرائح ROM هو الخط $\overline{OE}$ والذي عندما يكون فعالا يسمح بخروج البيانات المطلوب قراءتها على مسار البيانات لهذه الشريحة وبالتالي يستطيع المعالج قراءتها . هذا الطرف $\overline{OE}$ يوصل في العادة بالإشارة $\overline{MEMR}$ والذي يكون فعالا عندما يكون كلا من الخطين $\overline{RD}$ و $IO/\overline{M}$ صفرا .

شكل (16-4أ) يبين كيفية توصيل شرائح ROM على المعالج . بالنسبة لشرائح ذاكرة القراءة والكتابة RAM يكون لها ثلاث خطوط تحكم ، أولها يكون الخط $\overline{CS}$ أو $\overline{CE}$ كما ذكرنا سابقا ، وثانيها يكون الخط $\overline{OE}$ كما هو موجود في شرائح ROM ، وأما الخط الثالث فهو الخط $\overline{WE}$ أو خط تنشيط الكتابة Write Enable والذي يجب أن يكون فعالا عند التسجيل أو الكتابة في شريحة RAM . هذا الخط يوصل عادة بخط التحكم $\overline{MEMW}$ المكون من الخطين $\overline{WE}$ و $IO/\overline{M}$ القادمان من المعالج . شكل (16-4ب) يبين ذلك .

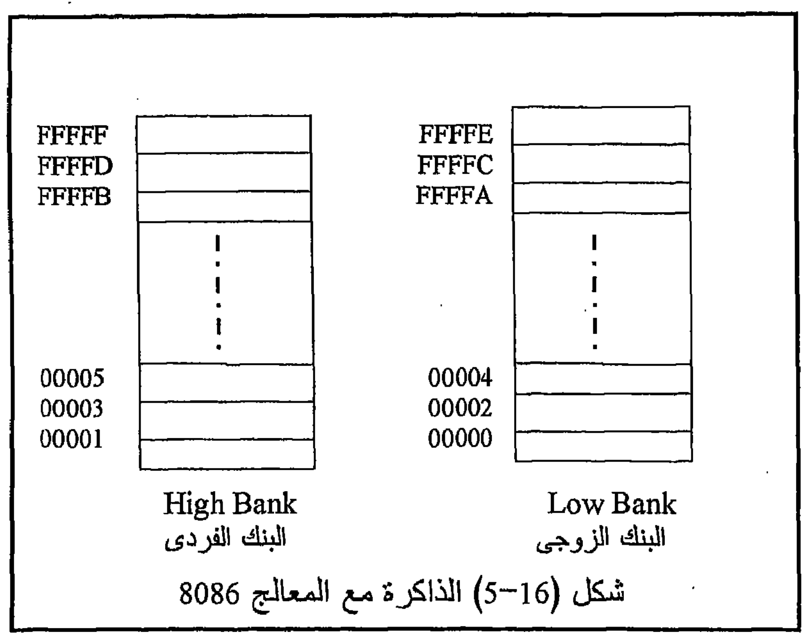
إن مواجهة المعالج 8086 تختلف كثيرا عن مواجهة المعالج 8088 وكل المعالجات السابقة ، حيث أن المعالج 8086 يكون مسار البيانات فيه 16بت ، ويكون هناك أوامر تتعامل مع هذا المسار على أساس 16 بت وأوامر أخرى تتعامل مع مسار البيانات على أساس 8 بت . لذلك فإن عملية المواجهة هنا يجب أن تأخذ ذلك في الحسبان . مسار العناوين للمعالج 8086 يتكون من 20 بت و لذلك فإنه سيتعامل مع ذاكرة مقدارها 1 ميجابايت إذا كان سيتعامل على مستوى البايت . أما إذا تعامل على مستوى 16بت (الكلمة) فإنه سيتعامل مع نصف هذه الكمية ، أي نصف ميجاوورد أو 512 كيلوورد . يتم ذلك بالطبع من خلال توصيلة خاصة للذاكرة مع كل من مساري البيانات والعناوين .

المعالج 8086 ينظر للذاكرة على أنها مقسمة نصفين أو بنكين . البنك الأول هو مجموعة البيانات ذات العناوين الزوجية ، والبنك الثاني هو مجموعة البيانات ذات العناوين الفردية كما في شكل (16-5) . البنك الأول أو الزوجي يسمى البنك الأدنى Lower Half لأنه يحمل نصف المعلومة الأدنى D0-D7 ، ويتم تنشيط هذا النصف باستخدام خط العناوين A0 الذي يكون صفرا عند التعامل مع أي عنوان زوجي حيث أن أي رقم زوجي تكون أول بت فيه تساوي صفرا . أما البنك الأعلى من المعلومة D8-D15 فيتم تخزينه في البنك الأعلى من الذاكرة Higher half ويستخدم في ذلك خط التحكم $\overline{BHE}$ الذي يكون فعالا (0) عند التعامل مع أي بايت في البنك العلوي . إذن معنى ذلك أن الخط A0 سيكون صفرا في حالة التعامل مع النصف الأدنى من الذاكرة على أساس 8 بت فقط ، ويكون أيضا صفرا في حالة التعامل مع الذاكرة على أساس 16 بت . أما الخط

$\overline{BHE}$ فيكون فعالا (0) في حالة التعامل مع البنك الأعلى من الذاكرة على أســـاس 8 بت ، أو التعامل مع الذاكرة على أساس 16بت .



شكل (16-4) مواجهة الذاكرة

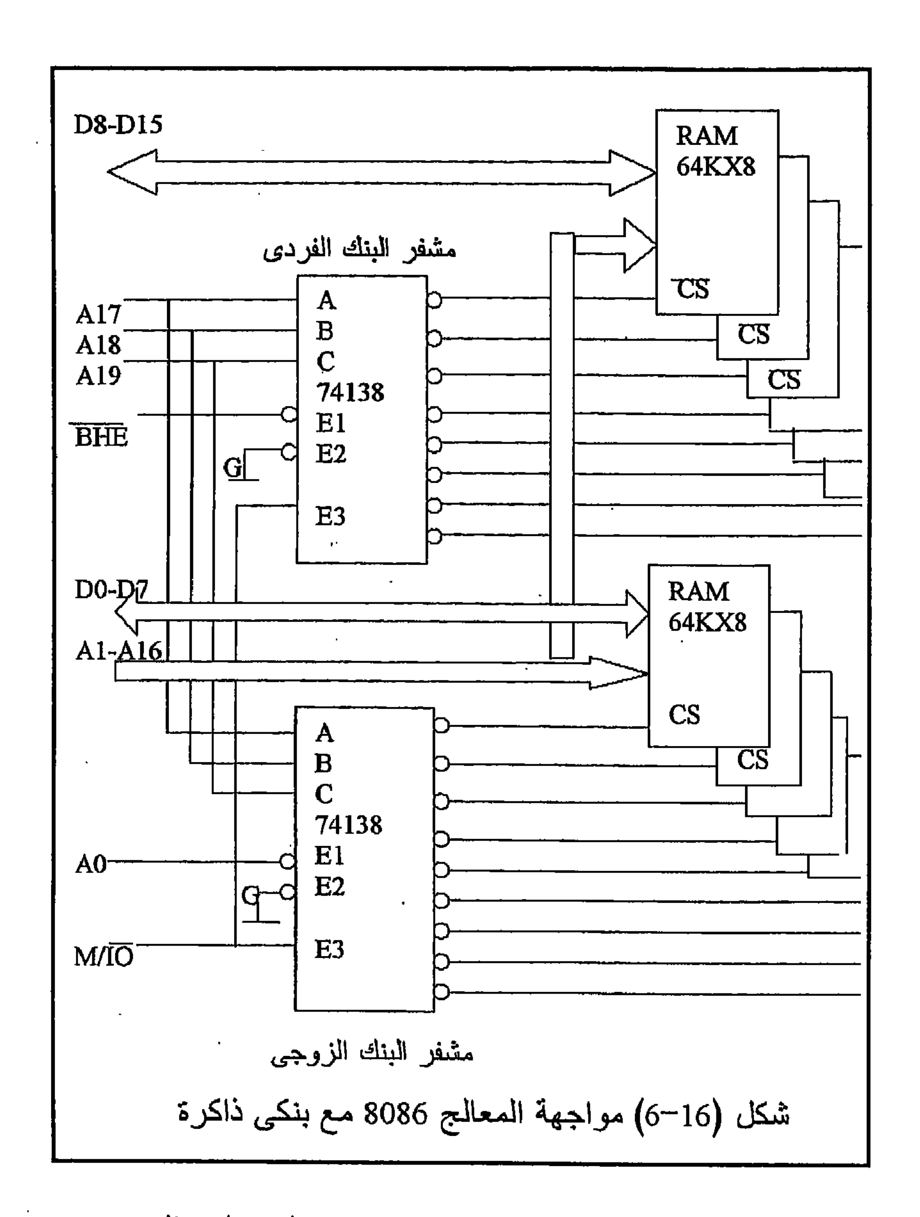


شكل (16-5) الذاكرة مع المعالج 8086

شكل (6–16) يبين كيفية استخدام مشفرين للعناوين للتعامل مع ذاكرة مقدارها 1 ميجابايت مقسمة إلي نصفين زوجي وفردي ، وكل نصف مكون من 8 شرائح كل منها 64 كيلوبايت . لاحظ ان كل من المشفرين تم توصيل خطوط التحكم لهم A, B, C على خطوط العناوين A17, A18, A19 على التوالي . أما خط التنشيط E3 في كل من المشفرين يتم توصيله على الخط $M/\overline{IO}$ والذي يكون واحد عند التعامل مع الذاكرة على العكس من الشريحة 8088 التي يكون فيها هذا الخط صفرا عند التعامل مع الذاكرة ، ولذلك فإن هذا الخط يرمز له بالرمز $IO/\overline{M}$ في حالة الشريحة 8088 . أما خط التحكم $\overline{E2}$ فتم توصيله بالأرضي في الشريحتين ليكون فعالا دائما ، وأما خط التحكم $\overline{E1}$ في المشفر الأول فمتصل بخط العناوين A0 ليختار النصف الأدنى أو الزوجي من الذاكرة ، والخط $\overline{BHE}$ تم توصيله على الخط $\overline{E1}$ في المشفر الثاني الذي يختار النصف الأعلى أو الفردي من الذاكرة . لذلك فإنه عند التعامل مع الذاكرة على أساس 16 بت فإن كل من A0 و $\overline{BHE}$ يكونان فعالان (0) ويتم التعامل مع نصفي الذاكرة في نفس الوقت حيث يتم قراءة أو كتابة البيانات على مسار البيانات بالكامل D0-D15 .

أما عند التعامل مع أحد النصفين الزوجي أو الفردي فإنه إما أن يكون الخط A0 فعالا (0) أو يكون الخط $\overline{BHE}$ فعالا (0) وبذلك نضمن التعامل مع أحد النصفين في حالة التعامل على أساس 8 بت .

في شكل (6–16) نلاحظ أن الخط A0 تم توصيله كخط تنشيط لمشفر النصف الزوجي من الذاكرة ، ثم بعد ذلك تم توصيل الخط A1 من المعالج بالخط A0 في الذاكرة ، والخط A2 من المعالج بالخط A1 في الذاكرة ، و بذلك نضمن التعامل مع كل بايت في الذاكرة في حالة التعامل على أساس 8 بايت ؛ أي سيتم استخدام جميع الواحد ميجابايت من الذاكرة . البعض سيقول لماذا لم تستخدم الخط A0 من المعالج على الخط A0 في الذاكرة وفي نفس الوقت نستخدمه على التوازي لتنشيط المشفر وباقي خطوط العناوين يتم توصيلها كل مع ما يناظره . حاول أن بتخيل هذا التوصيل وتابع عملية التعامل مع عناوين الذاكرة ابتداء من العنوان 0000 ثم العنوان 0001 ثم العنوان 0002 وهكذا ستجد أن عملية التخزين ستتم في بايت وتترك الأخرى مما يعني أن التعامل سيتم مع نصف كمية الذاكرة فقط ولن يمكن استخدام النصف الآخر منها ، ولذلك يجب تجنب استخدام هذه الطريقة للتوصيل .



شكل (16-6) مواجهة المعالج 8086 مع بنكى ذاكرة

16-5 الإدخال والإخراج من وإلي المعالج 8086/8088

لقد سبق شرح عملية الإدخال و الإخراج من وإلي المعالجات Z80 و 8085 في فصل سابق ورأينا أن هناك طريقتان للإدخال و الإخراج . الطريقة الأولى هي باستخدام الأمرين IN وOUT وسميت هذه الطريقة بطريقة الإدخال والإخراج

المباشر ، كما أن هناك طريقة أخرى للإدخال والإخراج وهي استعمال عناوين الذاكرة في ذلك . وهذا ما أسميناه بطريقة خرائط الذاكرة للإدخال والإخراج . كلا الطريقتين سنستخدمهم أيضا مع المعالج 8086 ولذلك ننصح بمراجعة هذا الفصل بالكامل ، لأننا سنبني على ما به من معلومات .

16-5-1 أوامر الإدخال والإخراج للمعالج 8086/8088

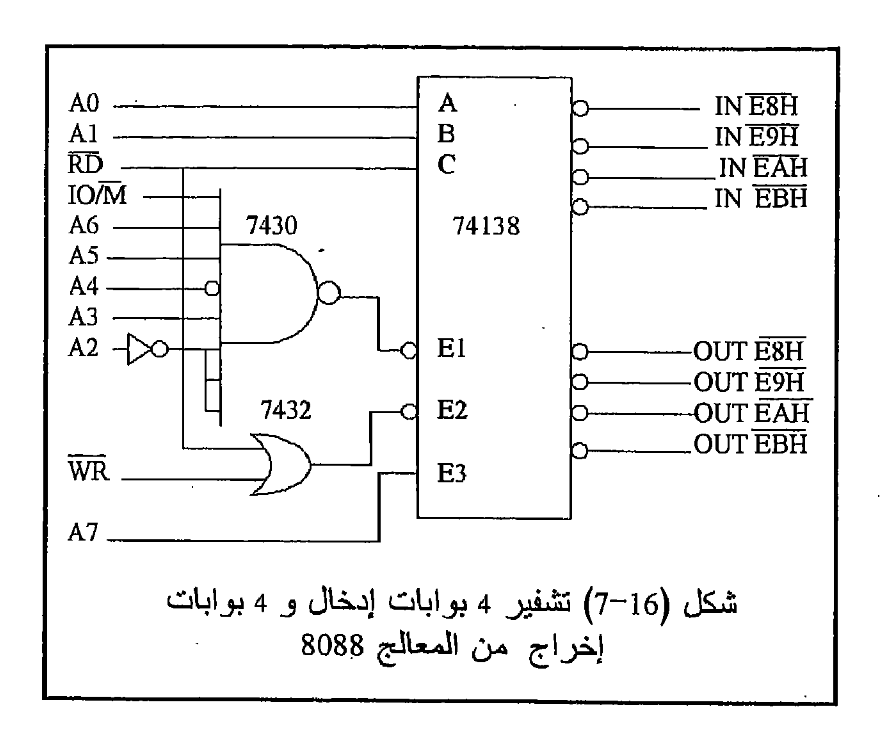
جدول 16-4 يبين جميع الحالات الممكنة لأوامر الإدخال والإخراج للمعالج 8086/8088 نلاحظ من هذا الجدول ما يلي :

- عنوان أي بوابة من الممكن أن يكون 8 بت أو 16 بت بحيث عندما يكون العنوان 16بت فإنه يوضع في المسجل DX . أما إذا كان العنوان 8 بت فإنه يوضع مباشرة في الأمر ، فمثلا الأمر IN AL,DX سوف يقرأ محتويات البوابة التي عنوانها موجودا في السجل DX (16بت) ويضعها في السجل AL . يمكن أيضا قراءة بوابة مكونة من 16بت كما في الأمر I N AX,0005 حيث سيقرأ البوابة رقم 0005 المكونة من 16بت ويضعها في المسجل AX (16بت) . بالطبع فإن الأوامر التي تتعامل مع بوابات 16بت ستكون محققة فقط مع المعالج 8086 حيث أن له مسار بيانات خارجي من 16 بت كما نعلم .
- الأمر I N AX,DX سيقرأ محتويات البوابة المكونة من 16 بت والتي يوجد عنوانها في المسجل DX ويضع هذه المحتويات في المسجل AX . لاحظ أنه طالما أن عنوان البوابة مكون من 16 بت فإن ذلك يعني أنه من الممكن التعامل مع 64ك من بوابات الإدخال و الإخراج وهذا كما نرى كم هائل من بوابات الإدخال والإخراج إذا ما قورن بالمعالج 8085 أو المعالج Z80 .
- جميع أساسيات الإدخال والإخراج التي درسناها في الفصول السابقة مطبقة هنا من حيث أنه لابد من عملية تشفير لعنوان البوابة ، كما أن عملية إدخال البيانات لابد وأن تكون من خلال عازل Buffer ثلاثي المنطق مثل الشريحة 74374 . عملية تشفير البوابات لابد أن تأخذ في الحسبان الطرف $IO/\overline{M}$ أو $M/\overline{IO}$ والذي يحدد متى يتعامل المعالج مع بوابات إدخال أو إخراج .

شكل (16-7) يبين مشفر لثمان بوابات ، أربعة منها للإدخال وأربعة للإخراج باستخدام المشفر 74138 . هذا المشفر يستخدم مع المعالج 8088 حيث أنه ثمان بتات فقط ، ونلاحظ أن عملية التشفير هنا هي نفسها عملية التشفير التي كنا نستخدمها مع المعالجات 8085 أو Z80 . لاحظ استخدام خطوط التحكم $IO/\overline{M}$ و $\overline{WR}$ و $\overline{RD}$.

الأمـــــــر	عرض البيانات	تعلـــــــيق
IN AL,d8	8	قراءة بوابة 8 بت عنوانها 8 بت (d8)
IN AL,DX	8	قراءة بوابة 8بت عنوانها فى (DX)
IN AX,d8	16	قراءة بوابة 16بت عنوانها 8بت (d8)
IN AX,DX	16	قراءة بوابة 16بت عنوانها فى (DX)
OUT d8,AL	8	كتابة فى بوابة 8بت عنوانها 8بت (d8)
OUT DX,AL	8	كتابة فى بوابة 8بت عنوانها فى (DX)
OUT d8,AX	16	كتابة فى بوابة 16بت عنوانها 8بت (d8)
OUT DX,AX	16	كتابة فى بوابة 16بت عنوانها فى (DX)

جدول 16-4



شكل (16-7) تشفير 4 بوابات إدخال و 4 بوابات إخراج من المعالج 8088

بما أن المعالج 8086 يختلف في طريقة تعامله مع الذاكرة عـــن المعـــالج 8088 نتيجة الاختلاف في مسار البيانات فإن هذا ينعكس أيضا على طريقـــة الإدخـــال والإخراج للبيانات وبالذات في حالة إدخال أو إخـــراج بيانـــات مـــن 8 بـــت ، وستكون المشكلة في هذه الحالة هي هل ستستخدم النصف الأدنـــى مـــن مســـار

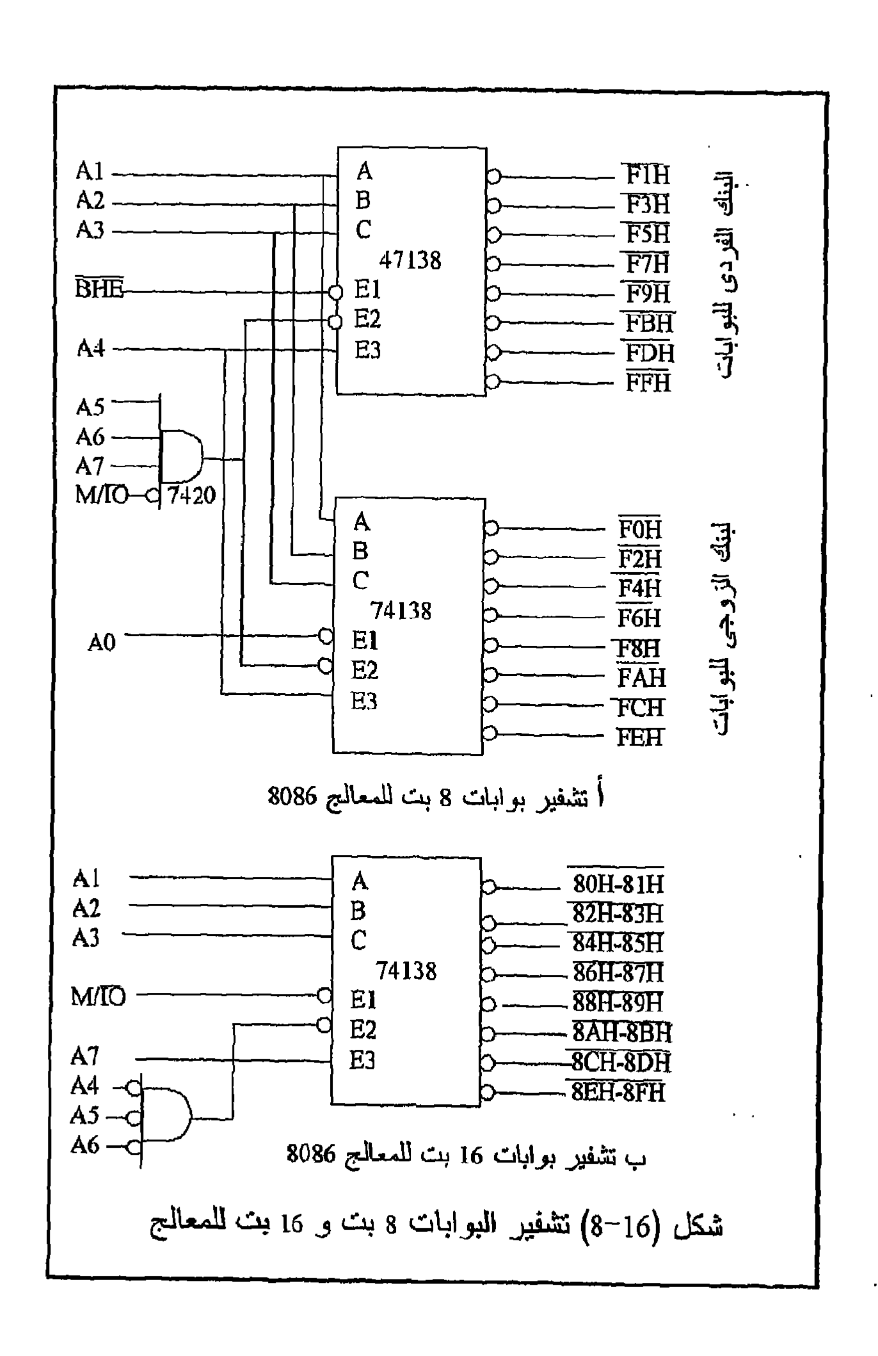
البيانات أم النصف الأعلى ، أم النصفين معا في حالة إدخال أو إخـــراج بيانـــات من 16 بت حيث في هذه الحالة لن تكون هناك أي مشكلة ، المشكلة فقط هي فـــي حالة التعامل مع بيانات وبوابات من 8 بت . إن أســـهل الطـــرق لتجنـــب هـــذه المشكلة هي التعامل إما مع النصف الأدنى فقط وفي هذه الحالة نجعل الخط A0 فعالا دائما (0) ؛ أو النصف الأعلى فقط وفي هذه الحالة نجعل الخط $\overline{BHE}$ فعالا (0) . شكل (16–8أ) يبين ذلك حيث تم استخدام مشفرين 74138 أحدهما يشـــفر لثمان بوابات بأرقام زوجية ....,F0, F2, F4 والآخر يشفر لثمان بوابات فرديـــة ...,F1, F3, F5 . شكل (16–8ب) يبين كيفية توصيل المشـــفر 74138 لتشـــفير ثمان بوابات ذات 16 بـــت عناوينـــها هـــي 80-81 و 82-83 و 84-85 ، ... . لاحظ عدم استخدام الخطين A0 و $\overline{BHE}$ في عملية التشفير حيث تم الاســـتغناء عنهما في هذه الحالة . في هذه الحالة مجرد وضع العنوان 80H مثـــلا سينشـــط الخرج الأول من المشفر وبالتالي ينشط نصفي البوابة 16 بت الملحقة بهذا الخـــط معا وفي نفس الوقت .

16–5–2 البوابات القابلة للبرمجة PPI

لقد سبق دراسة الشريحة 8255 في فصل سابق و كيفية مواجهتها مع المعالجـات ذات 8 بت والأمر لا يختلف كثيرا هنا ؛ لذا نحيل القارئ لمراجعة هذا الفصل . هناك بعض الشرائح الأخرى القابلة للبرمجة والكثيرة الاستخدام بالذات في مجـــال الحاسبات و سنعطي هنا فكرة سريعة عن بعض هذه الشرائح دون الدخول فـــي تفاصيل مواجهة هذه الشرائح . في حالة احتياج القارئ لتفاصيل أكثر عن هـــذه الشرائح فإننا نحيله إلي الكتالوجات الخاصة بهذه الشرائح والمراجع الموجودة في آخر الكتاب .

16–6 شريحة مواجهة لوحة المفاتيح القابلة للبرمجة 8279

الشريحة 8279 هي شريحة قابلة للبرمجة يمكن بها مواجهـــة لوحـــة المفـــاتيح وكذلك إظهار البيانات التي يتم إدخالها من هذه اللوحة . يمكن توصيل حتـــى 64 مفتاح على هذه الشريحة ، وتحتوي الشريحة على عازل buffer يمثل طـــابورا يسمح بتخزين 8 حروف من لوحة المفاتيح ثم استدعاء هذه الحروف على أســـاس من يصل أولا يخرج أولا عن طريق المعالج . جزء الإظهار يقـــوم بمســـح 16 مكان من أماكن الذاكرة التي تحتوي شفرات البيانات الثمانية المطلوب إظهارها .



أ تشفير بوابات 8 بت للمعالج 8086

ب تشفير بوابات 16 بت للمعالج 8086

شكل (16-8) تشفير البوابات 8 بت و 16 بت للمعالج

16-7 المؤقت القابل للبرمجة
Programmable Interval Timer, PIT 8254

تحتوي هذه الشريحة على ثلاث عدادات كل منها 16 بت ، وكل منها قابل للبرمجة . كل واحد من هذه العدادات قادر على العد الثنائي أو العد العشري المكود ثنائيا BCD وبترددات تصل إلي 10 ميجاهيرتز . يمكن استخدام هذه الشريحة في العديد من التطبيقات مثل الساعة الحقيقية Real time clock ، عدادات الأشياء ، التحكم في سرعة موتور واتجاهه مثل موتور الأسطوانة الصلبة أو حتى أي موتور ، والكثير من التطبيقات الأخرى التي يكون الزمن فيها عاملا مهما .

إن الحصول على أزمنة تأخير يمكن أن يتم باستخدام بعض أوامر أي لغة من لغات البرمجة ، فمثلا في لغة الأسمبلي يمكن الحصول على زمن تأخير باستخدام حلقة مغلقة كالتالي :

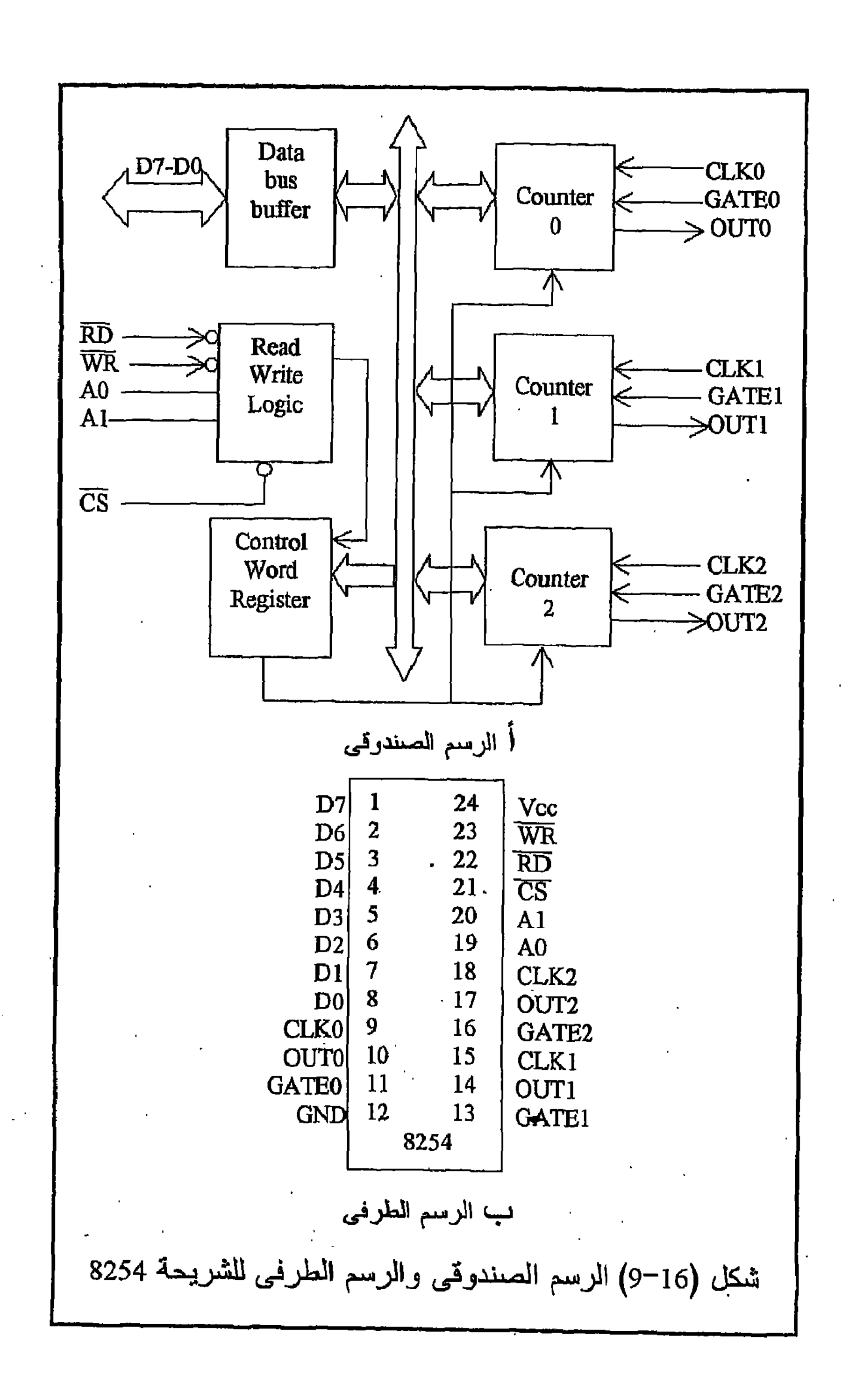
```
MOV CX,0055H
XX : NOP
LOOP XX
```

حيث سيتم تنفيذ هذه الحلقة عدد من المرات مقداره العدد الموجود في المسجل CX . والمعروف أن كل واحد من هذه الأوامر ينفذ في عدد معين من نبضات الساعة الخاصة بالجهاز ، بحيث يمكن معرفة مقدار زمن التأخير بالضبط بمعرفة تردد إشارة نبضات الساعة الخاصة بالجهاز Clock . من المعروف أن نبضات الساعة Clock تختلف من جهاز لآخر ولذلك فإن زمن التأخير الناتج عن الحلقة السابقة لن يكون ثابتا بأي حال باختلاف جهاز الحاسب الذي تنفذ عليه هذه الحلقة . إن استخدام شرائح التوقيت مثل الشريحة 254 8 يمكن بها الحصول على أزمنة تأخير ثابتة ولن تتوقف على نوع الجهاز الذي تستخدم معه لأنها تعمل على أساس نبضات ساعة ثابتة يتم توصيلها على هذه الشريحة كما سنرى .

شكل (16-9أ) يبين رسما صندوقيا لمحتويات هذه الشريحة وشكل (16-9ب) يبين رسما طرفيا لها . الرسم الصندوقي يوضح كيف أن هذه الشريحة مقسمة إلي جزأين أساسيين ، جزءا يواجه المعالج ، ويتكون من ثلاث أجزاء :

الأول : وهو عبارة عن عازل لمسارات البيانات D0 – D7 ويخرج منه 8 أطراف توصل على مسار البيانات (8 بت القادم من المعالج) .

الثاني : ويحتوي خطوط التحكم في القراءة و الكتابة وهي كالتالي :



أ الرسم الصندوقى

ب الرسم الطرفى

شكل (16-9) الرسم الصندوقى والرسم الطرفى للشريحة 8254

- الطرف $\overline{RD}$ الذي يوصل على خط القراءة القادم من المعالج ، حيث عندما يكون هذا الطرف فعالا (0) فإن المعالج يستطيع القراءة من المسجلات الموجودة داخل المؤقت .
- الطرف $\overline{WR}$ الذي يوصل على خط الكتابة القادم من المعالج ، حيث عندما يكون هذا الخط فعالا (0) فإن المعالج يستطيع كتابة أو إرسال بيانات إلي المسجلات الموجودة داخل المؤقت .
- الطرف $\overline{CS}$ ، لكي يمكن للمعالج أن يتعامل مع الشريحة PIT فإن الطرف $\overline{CS}$ وهو خط اختيار الشريحة Chip Select لابد وأن يكون فعالا (0) ، ويكون ذلك بالطبع عن طريق تشفير العنوان الخاص بهذه الشريحة كما رأينا عند التعامل مع الشريحة 8255 .
- الطرفان A0, A1 : هذان الطرفان يوصلان فى العادة على خطى العناوين A0, A1 القادمين من المعالج حيث يتم عن طريق هذين الخطين اختيار أحد عدادات الشريحة أو مسجل التحكم داخل الشريحة تبعا للجدول 16-5 .

الوظيفة	A0	A1
العداد رقم (0)	0	0
العداد رقم (1)	1	0
العداد رقم (2)	0	1
مسجل التحكم	1	1

جدول 16-5

الثالث: هو مسجل التحكم Control Register والذى يتم فيه تسجيل كلمة تحكم Control Word من 8 بتات تمثل اختيار أحد العدادات الثلاثة ليتم التعامل معه حسب حالة تشغيل معينة من خمس حالات تشغيل سنراها بعد قليل .
الجزء الثانى أو الجانب الآخر من الشريحة PIT كما فى شكل (16-9أ) يمثل الجانب المواجه للمستخدم ، وهو يمثل الثلاث عدادات الموجودة داخل الشريحة ، حيث لكل عداد منها ثلاث إشارات أو ثلاثة أطراف كما يلى :

- الأطراف CLKX حيث X تمثل رقم العداد (2, 1, 0) ويتم إدخال نبضات الساعة Clock التى سيقوم العداد بعدها على هذه الأطراف ، ولابد أن يكون تردد هذه النبضات معروفا جيدا ، ويمكن أن يصل هذا التردد حتى 10 ميجاهرتز .

- الأطراف GATEX وكل طرف منها يمثل طرف تنشيط للعداد المراد التعامل معه Gate Enable .
- الأطراف OUTX وتمثل أطراف خرج للعدادات الثلاثة ، ويمكن برمجة هذه العدادات ليكون الخرج واحدا أو صفرا أو نبضات على حسب حالة التشغيل التى يعمل عليها العداد كما سنرى .
- آخر طرفان من أطراف الشريحة هما طرفى القدرة GND, Vcc .

D7	D6	D5	D4	D3	D2	D1	D0
SC1	SC0	RW1	RW0	M2	M1	M0	BCD

SC1	SC0	العداد المختار
0	0	عداد رقم 0
0	1	عداد رقم 1
1	0	عداد رقم 2
1	1	قراءة حالة الشريحة

M2	M1	M0	الحالة
0	0	0	الحالة 0
0	0	1	الحالة 1
0	1	0	الحالة 2
0	1	1	الحالة 3
1	0	0	الحالة 4
1	0	1	الحالة 5

RW1	RW0	قراءة/كتابة
0	0	قراءة/كتابة أمر مسك العدادات
0	1	قراءة/كتابة النصف الأدنى فقط
1	0	قراءة/كتابة النصف الأعلى فقط
1	1	قراءة/كتابة النصف الأدنى ثم الأعلى

BCD	طريقة العد
0	عداد ثنائى
1	عداد عشرى مكود ثنائيا BCD

شكل (16-10) كلمة التحكم Control Word للشريحة 8254

16-7-1 برمجة الشريحة 8254

كما رأينا فإن الشريحة 8254 تحتوى ثلاث عدادات كل منها 16 بت ، أى يمكن لكل منها أن يعد من 0 إلى FFFFH فى حالة العد الثنائى ، أو العد من 0 إلى

9999 فى حالة العد العشرى . يمكن اختيار أى واحد من العدادات الثلاثة ، وطريقة التعامل معه ، وكذلك حالة التعامل عن طريق شفرة توضع فى مسجل التحكم . شكل (16–10) يبين مسجل التحكم ودلالة كل بت من بتات هذا المسجل. يحتوى شكل (16–10) أيضا على جداول توضح وظيفة كل مجموعة من مجموعات البتات فى هذا المسجل كما يلى :

- البتات D6 و D7 تمثل شفرات اختيار أحد العدادات Select Counter bits ليتم التعامل معه ، أو قراءة حالة الشريحة . فإذا كان كل من بت 6 و 7 تساوى صفر فإن المقصود فى هذه الحالة هو التعامل مع العداد رقم صفر ، أما إذا كانت بت 6 تساوى واحد ، وبت 7 تساوى صفر ، فإن التعامل فى هذه الحالة سيكون مع العداد رقم واحد ، وهكذا . أما إذا كان كل من بت 6 و7 تساوى واحد فإنه فى هذه الحالة سيتم قراءة مسجل التحكم .
- البتات D4 و D5 تمثلان كيفية القراءة أو الكتابة من أى واحد من العدادات الذى تم اختياره بالبتات 6 و 7 . كما نعلم فإن كل عداد مكون من 16 بت ، بينما مسار البيانات للشريحة مكون من 8 بت فقط ، لذلك فإنه لابد من تحديد أى بايت (8بت) من ال 16 بت سيتم قراءتها أو الكتابة فيها . فإذا كانت البت 4 تساوى واحد والبت 5 تساوى صفر فإنه فى هذه الحالة سيتم التعامل مع البايت الأولى Lower significant byte ، أما إذا كانت البت 4 تساوى صفر والبت 5 تساوى واحد فإنه سيتم التعامل مع البايت الثانية فى هذه الحالة Higher significant byte وأخيرا يمكن قراءة البايت الأولى ثم الثانية مباشرة بوضع كل من البت 4 و 5 تساوى واحد . قبل قراءة أى عداد فى أى لحظة لابد من مسك Latch قيمة العداد عند هذه اللحظة ووضعها فى مسجل القراءة . بذلك نضمن أنه فى أثناء قراءة أى بايت فإن البايت الأخرى لن تتغير فى أثناء القراءة . لذلك فإنه قبل قراءة أى عداد فإنه لابد من مسك محتويات هذا العداد بوضع البتات 4 و 5 كل منها تساوى صفر .
- البتات D1 و D2 و D3 يمكن عن طريقها اختيار الحالة mode التى سيعمل عندها العداد الذى تم اختياره . بهذه الثلاث بتات يمكن اختيار حالة من ست حالات يمكن لأى عداد أن يعمل عندها كما فى شكل (16–11) .
- البت رقم صفر D0 ويتم عن طريقها جعل العداد الذى يتم اختياره يعد عشرى أو ثنائى . فإذا كانت هذه البت تساوى صفرا فإن العداد المختار سيعد عدا ثنائيا من 0 حتى FFFFH ، أما إذا كانت هذه البت واحد فإن العداد سيعد عشريا من صفر حتى 9999 .

16-7-2 حالات تشغيل الشريحة PIT

الحالة 0

شكل (16-11) يبين الست حالات التى يمكن أن يعمل فيها أى عداد من العدادات الثلاثة الموجودة فى الشريحة 8254 . فى الحالة 0 وكما هو مبين فى شكل (16-11) فإن الخرج OUTX يكون واحد إلى أن يتم تحميل العداد رقم X بالرقم N ويتم تنشيط الخط GATEX حيث عندها ينزل الخرج OUTX إلى الصفر ، ويظل كذلك إلى أن ينتهى العداد X من عد N من نبضات الساعة حيث عند النبضة N+1 سيعود الخرج إلى الواحد مرة ثانية . فى هذه الحالة لابد أن يكون الخط GATEX نشط دائما .

الحالة 1

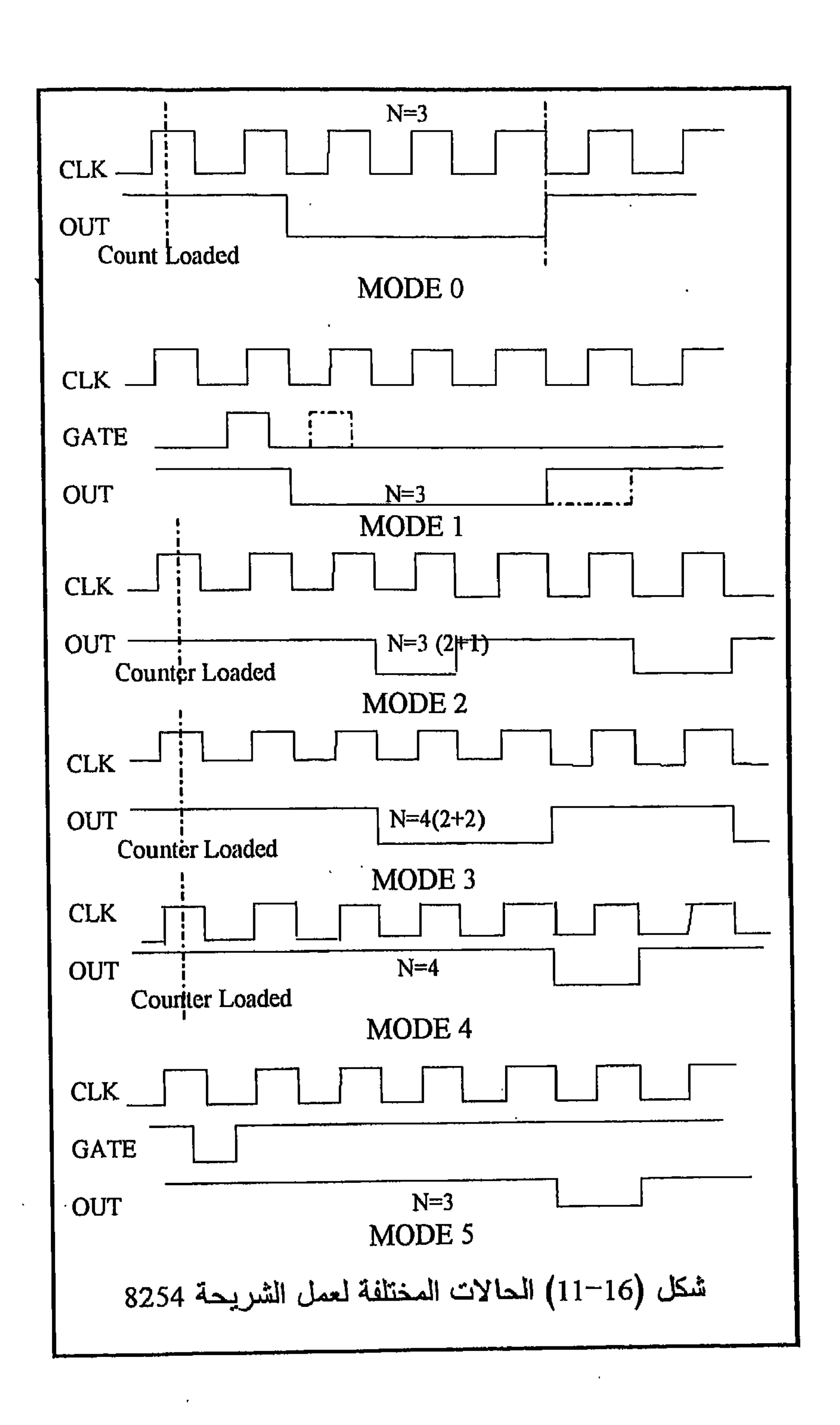
فى هذه الحالة سيعمل العداد كمؤقت أحادى الاستقرار يتم تنشيطه من الطرف GATEX حيث بإعطاء نبضة على هذا الخط فإن الخرج سينزل إلى الصفر ويظل كذلك إلى أن يصل العداد إلى القيمة المبرمجة حيث عندها يصعد الخرج إلى الواحد . لاحظ الفرق بين هذه الحالة والحالة السابقة والذى يقع فقط فى كيفية تنشيط الطرف GATEX . فى هذه الحالة إذا تم إعطاء نبضة تنشيط على الطرف GATEX أثناء نشاط الخرج ، فإن الخرج سيبدأ فترة نشاط جديدة كما فى الخطوط المنقطة فى شكل (16-11) .

الحالة 2

هذه الحالة يوضحها شكل (16-11) حيث يكون الخرج عديم الاستقرار . فى هذه الحالة يكون الخرج واحد طالما أن العداد لم يصل إلى القيمة المبرمج عليها ، وعندما يصل إلى هذه القيمة فإن الخرج ينزل إلى الصفر لمدة زمن نبضة تزامن واحدة ثم يرجع واحد ، وهكذا يتأرجح الخرج بين الواحد والصفر بتردد وأزمنة تأخير يتم التحكم فيها بالقيمة المخزنة فى العداد . لاحظ أن الطرف GATEX فى هذه الحالة لابد أن يكون فعالا .

الحالة 3

هنا يكون الخرج أيضا عديم الاستقرار حيث يكون عبارة عن موجة مربعة يتساوى فيها زمن الواحد وزمن الصفر وكل منهما له زمن يساوى نصف الزمن الناتج عن القيمة المبرمجة فى العداد إذا كانت هذه القيمة زوجية ، أما إذا كان العداد محمل بقيمة فردية فإن زمن الصفر يكون أقل بمقدار زمن نبضة تزامن واحدة عن زمن الواحد .



شكل (16-11) الحالات المختلفة لعمل الشريحة 8254

الحالة 4
بعد مرور عدد من النبضات مساوى للقيمة المحملة فى العداد ، فإن خرج العداد سينزل إلى الصفر لمدة زمن نبضة واحدة فقط بعدها يعود الخرج واحد كما كان ويظل كذلك إلى أن يتم برمجة العداد مرة أخرى . يمكن إخماد النبضة الناتجة عن طريق جعل الخط GATEX يساوى صفر .

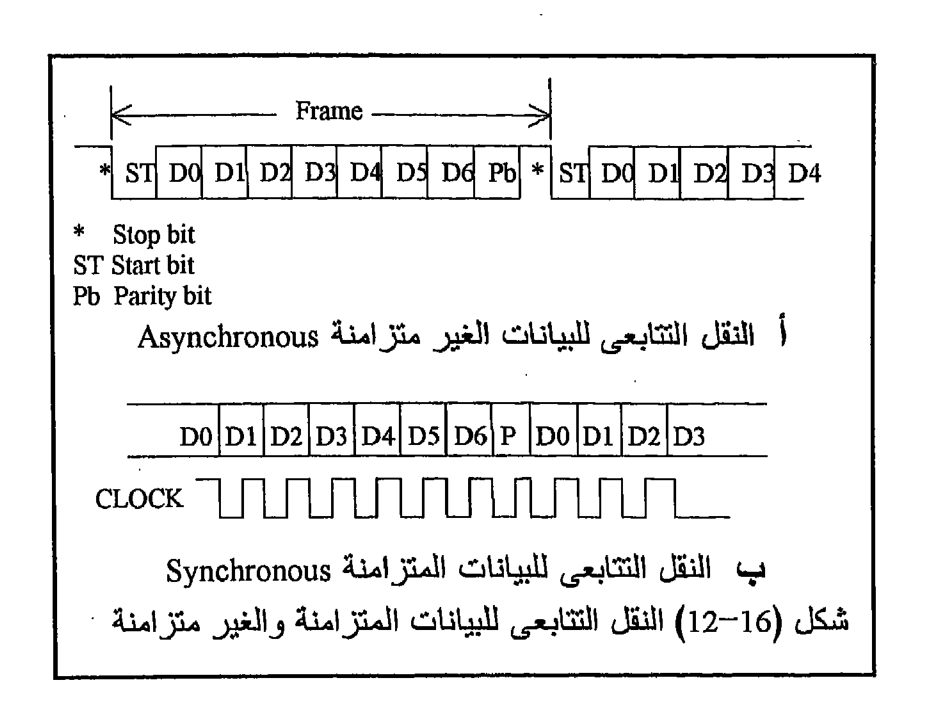
الحالة 5
هذه الحالة تشبه تماما الحالة 4 سوى أن زمن التأخير يبدأ عند إعطاء نبضة على الخط GATEX حيث بعد هذه النبضة بزمن يتحدد بالقيمة المبرمجة فى العداد ينزل الخرج للصفر لمدة زمن نبضة تزامن واحدة بعدها يرجع الخرج واحد مرة ثانية فى انتظار إعطاء نبضة أخرى على الطرف GATEX . لاحظ أن تنشيط زمن التأخير فى الحالة 4 يتم برمجيا (فقط يكون الطرف GATEX نشط) بينما فى الحالة 5 فإن زمن التأخير يتم تنشيطه بالطرف GATEX أى Hardware .

16-8 الاتصالات القابلة للبرمجة
Programmable Communication Interface, PCI 8251

الشريحة 8251 مصممة لتقوم بمواجهة نظم الاتصالات التتابعية مع المعالج . كما نعلم فإن عملية نقل البيانات إما أن تكون على التوازى ، أى أن المعلومة ترسل فى صورة بايت (8بت) كاملة على 8 خطوط إلى الهدف ، أو ترسل تتابعيا أى بت بعد بت على خط واحد مثل خط التليفون . فى العادة تستخدم الطريقة التتابعية عندما تكون المسافة بين المرسل والمستقبل كبيرة .

الشريحة 8251 عبارة عن مرسل Transmitter و مستقبل Receiver للبيانات الغير متزامنة Asynchronous أو البيانات المتزامنة Synchronous ، ولذلك يرمز لها بالاختصار USART والذى يعنى ما يلى :
Universal Synchronous Asynchronous Receiver Transmitter
وتتميز الشريحة بمعدلات إرسال baud rate عالية والتى تتمثل في عدد البتات التى يمكن إرسالها فى الثانية الواحد .
البيانات الغير متزامنة هى البيانات التى يتم إرسالها واستقبالها دون الحاجة إلى نبضات تزامن Clock . شكل (16-12أ) يبين إطارين من البيانات كل منهما 10 بت ، وكل منهما يحتوى على بت البداية Start bit ، وسبع بتات تمثل البيانات

المرسلة Data bits ، وبت للباريتى Parity bit وأخيرا بــت للنهايــة Stop bit ، ونلاحظ هنا عدم وجود نبضات تزامن مع هذه البيانات .



أ النقل التتابعى للبيانات الغير متزامنة Asynchronous

ب النقل التتابعى للبيانات المتزامنة Synchronous

شكل (16-12) النقل التتابعى للبيانات المتزامنة والغير متزامنة

إن مهمة الشريحة 8251 هى إضافة بتات البداية والنهايــة والبــاريتى للبيانــات المطلوب إرسالها ، ثم بعد ذلك تقوم بنقل هذه البتــات مــع البيانــات المطلــوب إرسالها على التتابع على خط الإرسال أو قناة الإرسال . عند المســتقبل توجـد شريحة أخرى من نفس النوع تقوم بالمهمة العكسية حيث تفصل البتات الإضافيـة عن البيانات الأساسية وتحسب الباريتى هل هى سليمة أم لا .

أما البيانات المتزامنة فلا تحتوى بتات إضافية بجانب بتات البيانات مثــل بتــات البداية والنهاية ولكن جميع البتات تمثل بيانات . كل بت من بتات البيانات لابــد أن تكوم متزامنة مع نبضة من نبضات التزامن كما فى شكل (16-12ب) ، أمــا بداية إطار البيانات فتحدد بحرف تزامن . لن نخوض فى تفاصيل هذه الشــريحة وطريقة برمجتها لقلة المتعاملين معها كشريحة منفصلة ولكن فــى العــادة يتــم التعامل معها كأحد مكونات نظام اتصالات متكامل .

16-9 الاتصال المباشر مع الذاكرة
Direct Memory Access, DMA 8237A

لقد رأينا فى طرق التعامل مع الأجهزة الخارجية كيف أنه لكى نخزن معلومة معينة من جهاز خارجى فى الذاكرة ، فإننا لابد أن نقرأ المعلومة أولا عن طريق المعالج ثم ننقلها بعد ذلك من المعالج إلى الذاكرة فى العنوان المحدد . أى أن المعالج لابد وأن يكون وسيط فى عملية نقل المعلومات من وإلى الذاكرة. مع تقدم الحاسبات وزيادة كمية البيانات التى يتم تداولها بين الأجهزة الخارجية والذاكرة الفعالة أو الأساسية ظهرت هناك فكرة تحرير المعالج من عملية الوساطة هذه بحيث تكون عملية نقل البيانات من الأجهزة المحيطة للذاكرة مباشرة ودون دخول المعالج كوسيط مما سيسرع من عملية نقل البيانات بدرجة كبيرة ، وهذا ما يطلق عليه الاتصال المباشر بالذاكرة . شكل (16-13) يبين رسما توضيحيا لهذه العملية . نلاحظ فى هذا الشكل وجود جهاز خارجى يتحكم فى هذه العملية وهو عبارة عن الشريحة 8237A والتى تمر من خلالها البيانات من وإلى المعالج دون أن تستقر فيها وإلا فقدنا ميزة السرعة . هذه الشريحة يتحدد دورها فى تحديد العناوين والغرض من التعامل مع الذاكرة هل هو القراءة.أم الكتابة . عندما يريد أى واحد من الأجهزة الخارجية مثل الاسطوانة الصلبة أن يتصل مباشرة بالذاكرة ، فإنه يطلب ذلك من المعالج عن طريق تنشيط الخط HOLD الداخل للمعالج بجعله يساوى واحد . عند ذلك وبعد الانتهاء من تنفيذ الأمر الحالى الذى ينفذه المعالج ، يقوم المعالج بالانفصال عن المسارات الثلاثة (البيانات والعناوين والتحكم) بجعلها جميعا فى حالة المقاومة العالية أو الحالة المنطقية الثالثة . بعد ذلك يخبر المعالج الجهاز الخارجى بأنه قد انفصل عن المسارات عن طريق تنشيط الخط HLDA بجعله يساوى واحد . عندما يشعر الجهاز الخارجى بذلك يفهم أن جميع المسارات أصبحت تحت سيطرته فيبدأ فى إرسال أو استقبال البيانات بمساعدة الشريحة 8237A . يظل المعالج منفصلا عن المسارات إلى أن يقوم الجهاز الخارجى بإخماد الخط HOLD إلى الصفر مرة أخرى حيث عندها يعود المعالج إلى السيطرة على المسارات مرة أخرى .

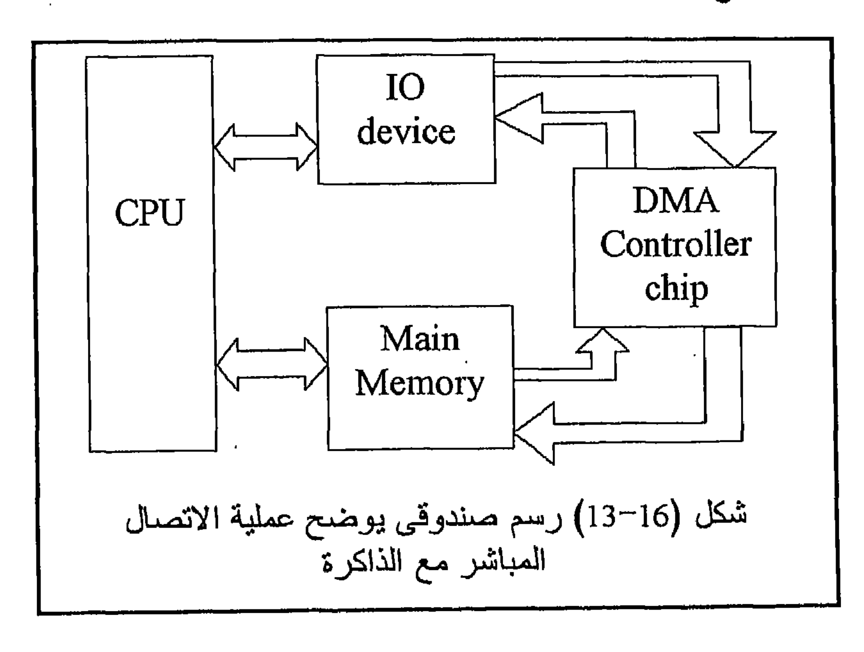
باستخدام الشريحة 8237A يمكن الاتصال المباشر بالذاكرة من خلال 4 قنوات اتصال . سنكتفى بهذا القدر من المعلومات عن هذه الشريحة لندرة استخدامها أيضا على المستوى الشخصى واستخدامها عادة فى الأنظمة المتكاملة مثل أنظمة الحاسب .

16-10 المواجهة مع المعالجات الحسابية المساعدة Arithmetic Coprocessor 80X87

عائلة المعالجات الحسابية المساعدة Arithmetic coprocessors هى عبارة عن معالجات تقوم بتنفيذ العمليات الحسابية والمقارنات بسرعة تفوق سرعة المعالج العادى حوالى 100 مرة وبالذات الدوال الحسابية مثل دوال حساب المثلثات ودوال الأسس وغيرها . بالإضافة إلى ذلك فإن هذه المعالجات تسهل بدرجة كبيرة التعامل مع البيانات المختلفة مثل الأرقام الصحيحة والحقيقية ذات الدقة المختلفة .

ابتداء من المعالج 8086 حتى المعالج 80386 سنجد أن كل منها له المساعد الحسابى الخاص به والذى يعمل معه ، فمثلا المعالج 8086 مساعده الحسابى هو الشريحة 8087 ، والمعالج 80186 مساعده الحسابى هى المعالج 80187 وهكذا .
ابتداء من المعالج 80486 بدأت شركة intell تضع كل معالج ومساعده الحسابى فى نفس الشريحة التكاملية بحيث أصبحت الأنظمة الحسابية لا تحتاج إلى المواجهة الخارجية مع المساعد الحسابى .

إننا لن نخوض أيضا فى تفاصيل مواجهة المعالجات المساعدة مع المعالج الأساسى لعدة أسباب منها ندرة استخدامها على المستوى الفردى ، وثانيا أن هذه المعالجات دخلت الآن ضمن مكونات المعالج العادى على نفس الشريحة بحيث أصبحت لا تنتج بصورة منفصلة .



شكل (16-13) رسم صندوقى يوضح عملية الاتصال المباشر مع الذاكرة

11-16 تمـــــارين

1. ما هو نوع الإشارة الموجودة على مسارى البيانات/العناوين حينمـــا يكـــون الخط ALE فعالا ؟
2. ما هو الغرض من خطوط الحالة S3 و S4 ؟
3. ما هى الحالة التى يكون فيها المعالج 8086/8088 حينما يكون الطـــرف $\overline{RD}$ يساوى صفرا ؟
4. اشرح الأطراف التالية للمعالج 8086/8088 :

- HOLD
- HLDA
- DT/$\overline{R}$
- $\overline{LOCK}$
- $\overline{TEST}$
- READY

5. ما هو الغرض من الطرف $\overline{BHE}$ ؟
6. لماذا نحتاج فى العادة لعملية فصل لمسارات أى معالج ؟
7. كيف نحدد اتجاه الإشارة على مسار البيانات عند استخدام الشـــريحة 74245 فى عملية عزل المسارات ؟
8. ما هو زمن الاتصال بالذاكرة ؟
9. ما هو الغرض من الطرف $\overline{DEN}$ ؟
10. ما هو الغرض من الطرف $\overline{CS}$ والطرف $\overline{OE}$ فى أى شريحة ذاكرة ؟
11. ارسم المشفر اللازم لعنونة المدى العنوانى DF800H-DFFFFH ؟
12. ارسم المشفر اللازم لعنونة المدى العنوانى 10000H-1FFFFH باســـتخدام 8 شرائح EPROM سعة كل منه 8 كيلوبايت .
13. أضف 8 شرائح RAM أخرى لنظام الذاكرة الموجود فى المسألة الســـابقة ، الشرائح سعة كل منها 2 كيلوبايت . ابدأ المدى العنوانى لـــهذه الشـــرائح عنـــد العنوان 20000H .
14. ما هو الغرض من الطرف A0 غير كونه خط عنونة ؟
15. اشرح كيف نحصـــل علـــى الخطـــوط $\overline{MEMR}$ و $\overline{MEMW}$ فـــى المعـــالج 8086/8088 .

الفصل السابع عشر

ثم ماذا؟

What else?

17-1 مقدمة

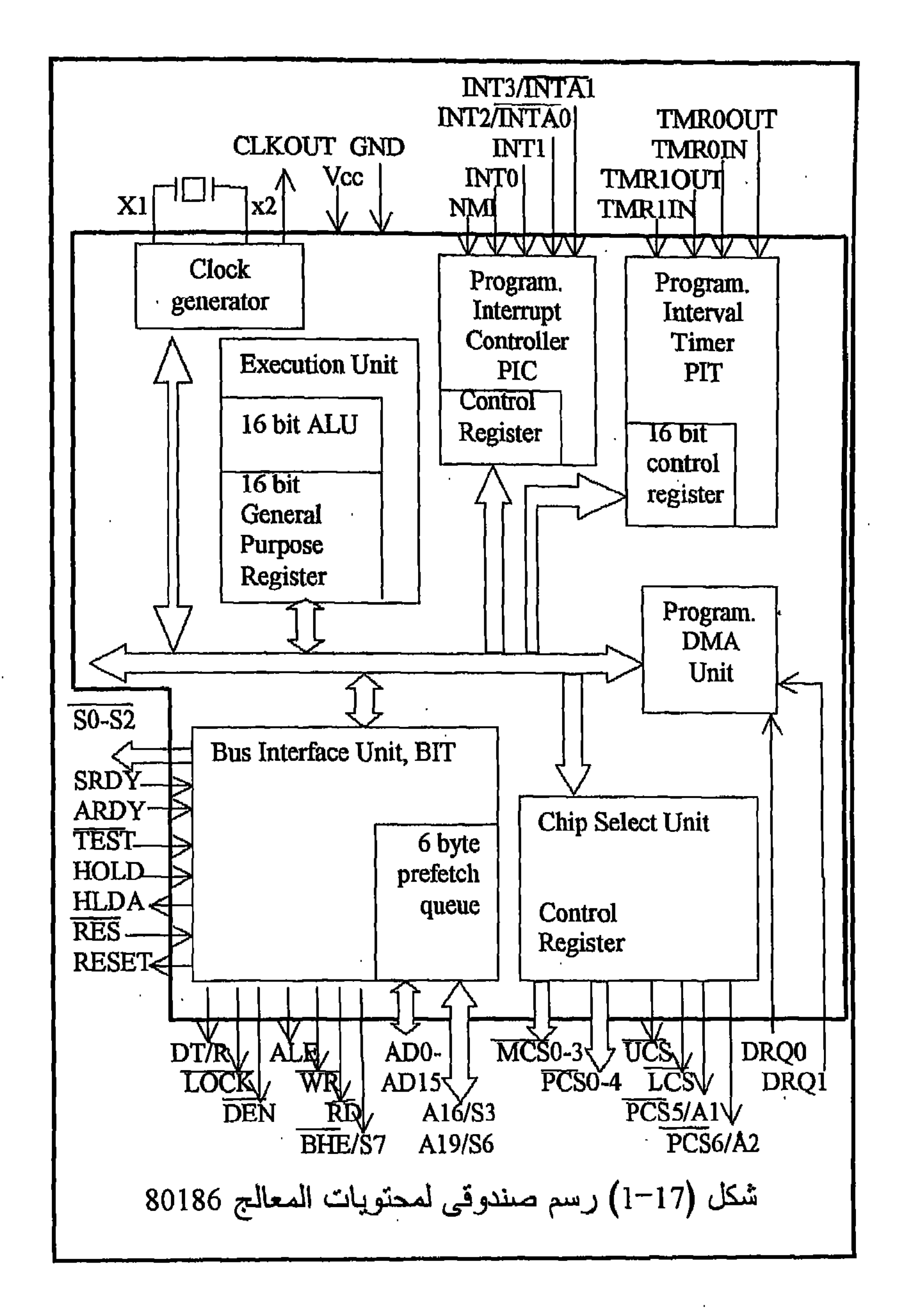
ثم ماذا بعد أن درسنا بالتفصيل المعالجين 8085 و Z80 كعينات من المعالجات ذات 8 بت والتى تتميز ببساطتها وسهولة برمجتها وسهولة مواجهتها مع الدوائر الخارجية ، ولذلك فإنها غالبا تكون هى المرشحة للاستخدام فى بناء دوائر التحكم التى نراها كثيرا فى التطبيقات الصناعية والكثير من الأجهزة الحديثة . ثم بعد ذلك درسنا بالتفصيل أيضا المعالج 8086/8088 كأحد المعالجات 16 بت والذى ، كما سنرى ، سيكون هو الأساس لكل المعالجات التالية التى سنراها فى هذا الفصل . ولذلك فإننا لن نخوض فى تفاصيل هذه المعالجات ولكننا سنكتفى بدراسة الإضافات والفروق التى تمت عليها . سنحاول بقدر الإمكان تغطية جميع المعالجات ابتداء من المعالج 80186 وانتهاء بالمعالج بنتيم برو Pentium Pro أحدث المعالجات فى الساحة الآن .

17-2 المعالج 80186

شكل (17-1) يبين رسما صندوقيا لمحتويات المعالج 80186 . هذا المعالج يشبه إلى حد كبير سابقه المعالج 8086 من حيث مسار البيانات الذى يتكون من 16 بت ومسار العناوين الذى يتكون من 20 بت . الجديد هنا هو أن الكثير من الشرائح الضرورية التى كان يستعملها المعالج 8086 وكانت توصل معه من الخارج ، تم إدخالها جميعها داخل شريحة المعالج نفسه وذلك لتبسيط دوائر المواجهة مع المعالج 80186 . المعالج 80186 له رفيق آخر وهو المعالج 80188 الذى يشبهه تماما فيما عدا أن مسار البيانات الخارجى يتكون من 8 بت بدلا من 16 بت . مازال كل من المعالجين أيضا يتكون من وحدتين أساسيتين وهما وحدة التنفيذ Execution Unit, EU ووحدة مواجهة المسارات Bus Interface Unit, BIU . شكل (17-1) يوضح البلوكات الأساسية التالية للمعالج 80186 :

1. وحدة نبضات الساعة Clock Generator

هذا المولد يحل محل الشريحة 8284A التى قدمناها فى فصل سابق والتى كانت توصل من خارج المعالج لتوفير نبضات الساعة وتوفير عمليات التزامن لكثير من إشارات التحكم مثل الطرف Ready .



شكل (17-1) رسم صندوقى لمحتويات المعالج 80186

هذا البلوك يخرج منه الطرفان X1 و X2 اللذان يوصل عليهما بللـــورة Crystal ذات تردد يساوى ضعف التردد المطلوب للمعالج ، فإذا كان المعالج سيعمل عنـد تردد 8 ميجاهيرتز مثلا فإن البلورة يجب أن يكـــون ترددهـــا 16 ميجـــاهيرتز . يخرج أيضا من هذا البلوك الطرف CLKOUT الذى يحمـــل نبضـــات الســـاعة الناتجة من داخل المعالج إلى خارجه حتى يمكن استعمالها بأى دائرة خارجية .

2. وحدة منظم المقاطعة القابل للبرمجة

Programmable Interrupt Controller, PIC

يحتوى المعالج 80186 على الشريحة 8259A التى تقوم بتنظيم عمليات المقاطعة حسب أولويات وصولها .هناك خمس مداخل لهذا البلوك وهى خطوط المقاطعة INT3, INT2, INT1, INT0 وخط المقاطعة الغير قابل للحجب Nonmaskable Interrupt NMI .

3. وحدة المؤقتات Timers

يحتوى هذا الجزء على ثلاث مؤقتات كل منها 16 بت وكلها قابلة للبرمجة مثل الشريحة 8254A والتى تناولناها فيما سبق بالتفصيل . كل هذه المؤقتات يمكنها أن تعمل إما على نبضات الساعة الداخلية الموجودة فى المعالج ، أو مع نبضات خارجية بأى تردد مطلوب .

4. وحدة الاتصال المباشر بالذاكرة

Direct Memory Access, DMA

يحتوى المعالج 80186 على وحدة اتصال مباشر بالذاكرة DMA ذات قناتى اتصال قابلة للبرمجة مشابهة تماما للشريحة 8237A .

5. وحدة اختيار الشرائح القابلة للبرمجة

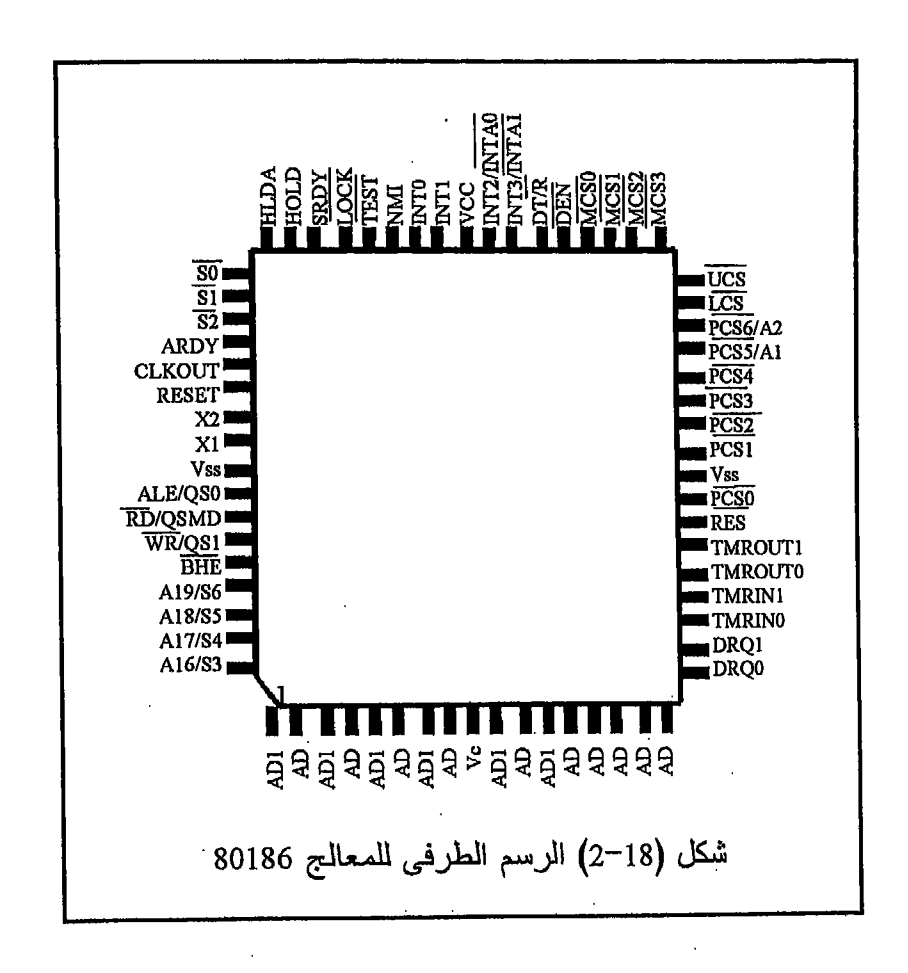
Programmable Chip Select Unit, PCS

هذه الوحدة عبارة عن مشفر قابل للبرمجة يوفر 6 خطوط لاختيار عناوين القاعدة base addresses أو عناوين البداية لمقاطع ذاكرة مختلفة ، كما توفر 7 خطوط لاختيار عناوين بوابات إدخال أو إخراج . فكر فى مدى ما يوفره مثل ذلك من التوصيلات الخارجية فى حالة تشفير هذه العناوين خارجيا .
يصدر المعالج 80186 فى شريحة مكونة من 68 طرفا فى شكل مربع مختلف عن كل الشرائح السابقة ذات 40 طرفا . شكل (17–2) يبين رسما طرفيا لهذا المعالج وفيما يلى سنعرض فكرة مبسطة عن وظيفة كل طرف من هذه الأطراف:

17–2–1 أطراف المعالج 80186

- الطرف Vcc وهو طرف القدرة لهذا المعالج ويساوى 5 فولت وهو الطرف رقم 9 فى الشريحة .
- الطرف Vss ويمثل الأرضى الخاص بالشريحة .

- الطرفان X1 و X2 وكما ذكرنا يوصلان على بللورة من الخارج للحصول على نبضات التزامن اللازمة . لاحظ أن تردد النبضات داخل المعالج يكون نصف تردد البلورة .
- الطرف CLKOUT تخرج عليه نبضات التزامن التى تم الحصول عليها حتى يمكن استخدامها بواسطة الأجهزة الخارجية .



شكل (2-18) الرسم الطرفى للمعالج 80186

- الطرف $\overline{\text{RES}}$ وهو طرف إعادة الوضع للمعالج Reset ويجب أن يظل هذا الطرف صفرا لمدة 50 ميللثانية حتى يتم إعادة الوضع . عند تنشيط هذا الطرف يذهب المعالج للعنوان FFFF0H لتنفيذ ما هناك من أوامر .
- الطرفان TMRIN0 و TMRIN1 يتم إدخال نبضات الساعة الخاصة بالمؤقتين 0 و 1 على هذين الطرفين .

• الطرف $\overline{TEST}$ ، يستخدم هذا الطرف بواسطة الأمر WAIT حيث أنه عندما يكون هذا الطرف فعالا (0) فإنه لن يكون هناك انتظار ، ولكى يتم الانتظار لابد أن يكون هذا الطرف واحد .
• الطرفان TMROUT0 و TMROUT1 وهما طرفا خرج تخرج عليهما إشارة خرج المؤقتين والتى تكون إما فى صورة موجة مربعة أو نبضة واحدة .
• الطرفان DRQ0 و DRQ1 وهما طرفا دخل يتم عليهما طلب الاتصال المباشر مع الذاكرة DMA من خلال القناتين 0 أو 1 وهما فعالان عندما يكون كل منهما 1 .
• الطرف NMI وهو طرف دخل ، تدخل عليه إشارة طلب المقاطعة الغير قابلة للحجب nonmaskable ، وهذا الطرف ينشط مع الحافة الصاعدة للإشارة .
• الأطراف INT0 و INT1 و INT2/$\overline{INTA0}$ و INT3/$\overline{INTA1}$ ، كلها أطراف دخل تدخل عليها إشارة طلب المقاطعة القابلة للحجب والتى أرقامها 0 و 1 و 2 و 3 وكلها فعالة عندما تكون 1 . هذه الخطوط يمكن برمجتها لتكون 4 خطوط طلب مقاطعة ، أو خطين لطلب المقاطعة وخطين للاعتراف acknowledge بهذه المقاطعة .
• الخطوط A16/S3 و A17/S4 و A18/S5 و A19/S6 ، عبارة عن 4 أطراف تستخدم فكرة المزج الزمنى بين إشارة العناوين A16 إلى A19 وخطوط الحالة S3 إلى S6 . خط الحالة S6 يبين إذا كان المعالج فى حالة اتصال مباشر مع الذاكرة حيث عندها يكون هذا الخط 1 ، ويكون صفرا فى حالة التشغيل العادى للمعالج . باقى خطوط الحالة تكون أصفارا .
• الخطوط AD0 إلى AD15 ، عبارة عن 16 خط تخرج عليها إشارة العناوين والبيانات فى مزج زمنى مثل المعالج 8086 .
• الطرف $\overline{BHE}$/S7 طرف خرج يبين إذا كانت الإشارة الموجودة على النصف العلوى من مسار البيانات تمثل بيانات محققة ، هذا الخط ممزوج زمنيا مع الإشارة S7 .
• الطرف ALE/QS0 وهو طرف خرج عبارة عن مزج زمنى بين إشارة تنشيط ماسك العناوين Address Latch Enable, ALE والإشارة QS0 والتى تمثل حالة طابور queue الأوامر فى وحدة مواجهة المسارات .
• الطرف $\overline{WR}$/QS1 ، خط خرج يبين إذا كان المعالج يكتب بيانات إلى الذاكرة أو وحدة إخراج . هذا الطرف ممزوج زمنيا مع الإشارة QS1 التى تمثل الإشارة الثانية لحالة طابور الأوامر .
• الخط $\overline{RD}$/QSMD خط خرج يبين إذا كان المعالج يقرأ من الذاكرة أو من وحدة إدخال . هذا الخط ممزوج زمنيا مع الخط QSMD أو خط بيان حالة الطابور Queue Status Mode .

- الطرف Asynchronous Ready, ARDY طرف دخل للمعالج يخبره إذا كانت الذاكرة أو وحدة الإدخال أو وحدة الإخراج جاهزة Ready . عندما يكون هذا الخط صفر يدخل المعالج في حالة انتظار .
- الطرف Synchronous ready, SRDY هذا الطرف مثله مثل الطرف ARDY فيما عدا أنه لابد و أن يكون متزامن مع نبضات الساعة الخاصة بالنظام . إذا كان هذا الخط صفر يدخل المعالج في حالة انتظار.
- الطرف $\overline{LOCK}$ خط خرج يبين إذا كان الأمر الذي يتم تنفيذه أمر محظور على المعالج المساعد أم لا ، حيث أنه يمكن إضافة بايت قبل أي أمر تمنع المعالج المساعد من الحصول على مسارات النظام ، وفي هذه الحالة يكون الطرف LOCK فعالا ويساوي صفرا .
- الخطوط $\overline{S0}, \overline{S1}, \overline{S2}$ أطراف خرج تمثل حالة المعالج أثناء أي عملية نقل للبيانات .
- الطرف HOLD طرف دخل يطلب من المعالج الانفصال عن المسارات لكي تتم عملية اتصال مباشر DMA مع أحد الأجهزة الخارجية .
- الطرف HOLDA طرف خرج يمثل إشارة اعتراف من المعالج بقبول الانفصال عن المسارات .
- الطرف Upper memory Chip Select, $\overline{UCS}$ طرف خرج يستخدم كخط اختيار لعناوين الذاكرة في الجزء العلوي من خريطة الذاكرة . يمكن برمجة هذا الطرف لاختيار من 1 كيلو بايت حتى 256 كيلو بايت تنتهي بالعنوان FFFFF .
- الطرف Lower memory Chip Select, $\overline{LCS}$ طرف خرج يستخدم كخط اختيار لعناوين ذاكرة في الجزء الأدنى من خريطة الذاكرة . أيضا يمكن برمجة هذا الخط لاختيار من 1 كيلو بايت حتى 256 كيلو بايت تبدأ بالعنوان 00000H.
- الأطراف Mid Memory Chip Select $\overline{MCS0}$-$\overline{MCS3}$ أربع أطراف خرج تستخدم كخطوط اختيار لعناوين الذاكرة في أي مكان في الخريطة . يمكن برمجة أي طرف لاختيار من 8 كيلو بايت وحتى 512 كيلو بايت تبدأ عند أي عنوان في الذاكرة .
- الأطراف $\overline{PCS0}$-$\overline{PCS4}$ خمس خطوط خرج تستخدم لعنونة أجهزة الإخراج والإدخال .
- الطرفان $\overline{PCS5}$/A1 , $\overline{PCS6}$/A2 خطوط خرج تستخدم إما لعنونة أجهزة الإخراج والإدخال مثل $\overline{PCS0}$-$\overline{PCS4}$ ، أو كخطوط عنونة A0, A1 .
- الطرف DT/$\overline{R}$ خط خرج يبين اتجاه البيانات على مسار البيانات إذا كانت خارجة أم داخلة للمعالج .
- الطرف $\overline{DEN}$ يستخدم لتنشيط فواصل مسار البيانات الخارجية حيث يكون هذا الخط فعال (0) في حالة وجود بيانات على مسار البيانات .

17-2-2 برمجة المعالج 80186

جميع أوامر الشريحة 8086 قابلة للتنفيذ دون أي مشاكل مع المعالج 80186 . يحتوي المعالج 80186 بعض الأوامر الإضافية التي لم تكن موجودة أصلا مع المعالج 8086 من هذه الأوامر ما يلي :

- ليس هناك أمر في المعالج 8086 يضرب قيمة فورية أو ثابت في محتويات أي مسجل ، فمثلا الأمر MUL BX, 2300H غير معرف مع المعالج 8086 ولكن مع المعالج 80186 يمكن ضرب أي قيمة فورية في محتويات أي مسجل باستخدام الأمر

IMUL BX, data 16

حيث سيضرب الثابت data16 في محتويات المسجل BX ويضع النتيجة في المسجل BX .

- الأمر SHL BX, 4 هذا الأمر يقوم بإزاحة محتويات المسجل BX لليسار بمقدار 4 أماكن أو 4 بتات . في المعالج 8086 كان هناك إمكانية للدوران أو الإزاحة بمقدار بت واحدة فقط .
- هناك بعض الأوامر الإضافية على عمليات الإضافة PUSH والسحب POP من المكدسة .

بالطبع لابد وأن يكون هناك أوامر إضافية للتعامل مع الشرائح الإضافية والتي أدخلت داخل شريحة المعالج مثل المؤقتات والاتصال المباشر مع الذاكرة والمقاطعة .

سنكتفي بذكر هذه الفروق في صورة عامة دون الدخول في تفاصيل وذلك لندرة البرمجة أو الحاجة لهذه الأوامر الإضافية .

17-3 المعالج 80286

المعالج 80186 لم يستمر كثيرا في السوق ولم يتعدى عمر خدمته في أنظمة الحاسبات سوى عام أو عامين على الأكثر حتى ظهر المعالج 80286 الذي كان بداية نقلة من الحاسبات XT إلي الحاسبات AT . المعالج 80286 عبارة أيضا عن امتداد للمعالج 8086 ويستطيع التعامل مع ذاكرة مقدارها 16 ميجا بايت نتيجة زيادة خطوط العناوين إلي 24 خطا بدلا من 20 خطا في حالة المعالج 8086 . هذا بالإضافة إلي وحدة جديدة وهي ما يسمى بوحدة إدارة الذاكرة Memory Management Unit, MMU التي بواسطتها يمكن التعامل مع كمية من الذاكرة التخيلية تصل إلي 1 جيجا بايت . هذا بالإضافة إلي أن المعالج 80286

يمكنه التعامل مع أكثر من مستخدم ولذلك يطلق عليه بأنه متعـــدد المســـتخدمين Multitasking أو Multi-user .

17-3-1 التركيب الهيكلي للمعالج 80286

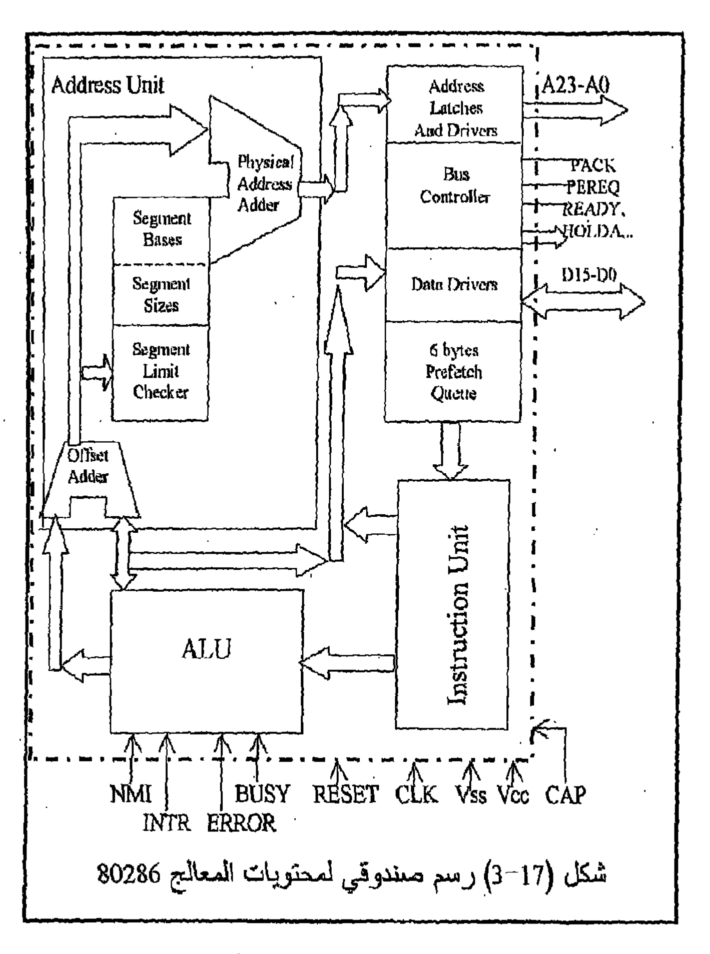
شكل (17-3) يبين رسما صندوقيا للمعالج 80286 حيث نلاحظ من هذا الشكل أن المعالج 80286 لا يحتوي شرائح المواجهة التي كـــانت موجـــودة فـــي المعـــالج 80186 ولكن بدلا من ذلك فإنه يحتوي على وحدة إدارة الذاكرة الجديدة MMU و التي يطلق عليها وحدة العنونة Address Unit في هذا الشكل .

يمكن للمعالج 80286 أن يعمل في واحدة من حالتين ، الحالـــة الأولـــى تســـمى الحالة الحقيقية real mode وفيها يكون المعالج 80286 مشـــابها تمامـــا للمعـــالج 8086 حيث يكون مسار العناوين في هذه الحالة 20 خطا فقط مما يسمح بعنونة 1 ميجا بايت ، أما باقي خطوط العناوين A20-A23 فتكون أصفارا في هذه الحالة ، وفي هذه الحالة فإن جميع برمجيات software الشريحة 8086 سوف تعمل مـــع المعالج 80286 بدون أي تعديل أو أي مشكلة .

الحالة الثانية أو الحالة الجديدة للمعالج 80286 تسمى الحالـــة المحميـــة التخيليـــة protected virtual mode وفي هذه الحالة فإن جميع خطوط مسار العنـــاوين A0-A23 تستخدم ، مما يتيح التعامل مع ذاكرة مقدارها 16 ميجا بـــايت . فـــي هـــذه الحالة يتم استخدام وحدة إدارة الذاكرة MMU التي تتيح عنونة حتـــى 16 كيلـــو جزء ؛ كل جزء مكون من 64 كيلو بايت أي أنها يمكنها عنونة حتـــى 16ك64Xك = 1 جيجا بايت من الذاكرة التخيلية .

بالإضافة لما تقدم ، تحتوى الشريحة 80286 على بعض المســـجلات الإضافيـــة الغير موجودة فى الشريحة 8086 المستخدمة فى وحدة إدارة الذاكـــرة . بـــالطبع فإنه نتيجة إضافة وحدة الذاكرة فلابد أن يكون هنـــاك مجموعـــة مـــن الأوامـــر الإضافية التى تستخدم للتحكم فى هذه الوحدة ، وهذه المجموعة هـــى الاختـــلاف الوحيد فى مجموعة الأوامر بين المعالج 80286 والمعالج 8086 .

يستخدم المعالج 80286 فكرة الذاكرة التخيلية virtual memory بحيـــث يمكـــن تخصيص جزء من الذاكرة لكل مستخدم user أو كـــل هـــدف task . يجـــب أن نتذكر جيدا أنه عندما يقوم المعالج بتنفيذ عدة برامج لأكثر من مستخدم أو أكـــثر من هدف على التوازى فإنه فى الحقيقة ينفذ جزء من البرنامج الأول الذى يكـــون ذو أولوية عالية ، وإذا انخفضت أولوية هذا الهدف نتيجة تنفيذ جزء منه ، فـــإن المعالج يتركه وينفذ فى الهدف الثانى أو الثالث ثم يرجع للهدف الأول وهكـــذا ، أى أن عملية التنفيذ تكون موزعة على الأهداف على التتابع ونتيجة السرعة فـــى تنفيذ هذه البرامج يشعر كل مستخدم كما لو كانت كل هذه الـــبرامج تنفـــذ علـــى التوازى .



شكل (3-17) رسم صندوقي لمحتويات المعالج 80286

من المشاكل الأساسية الموجودة في المعالج 80286 أنه عندما يدخل في الحالة المحمية التخيلية فإنه لا يستطيع الخروج منها والرجوع إلى الحالة الحقيقية real mode إلا إذا تمت إعادة وضع reset للمعالج ، وهذه بالطبع مشكلة كبيرة لأنها تأخذ وقتا كبيرا وتفقدك كل محتويات الذاكرة . هذه المشكلة تم التغلب عليها في المعالج 80386 .

17-4 المعالج 80386

كانت أول متطلبات هذا المعالج هى تطوير المعالج 80286 بحيث يمكن الرجوع من الحالة المحمية إلى الحالة الحقيقية بسهولة ، وقد كان ذلك حيث يمكن فى المعالج 80386 الانتقال من حالة لأخرى باستخدام أمر معين بدلا من إعادة وضع المعالج ويعتبر هذا إنجازا كبيرا .
الجديد أيضا فى المعالج 80386 أن مسار البيانات له مكون من 32 بت ، أى أنه يستطيع نقل 4 بايت كاملة من أو إلى الذاكرة أو أى جهاز خارجى فى رحلة واحدة فقط . كذلك فإن مسار العناوين لهذا المعالج مكون من 32 بت أيضا مما يتيح له التعامل مع ذاكرة مقدارها 4 جيجابايت . أما إذا دخل المعالج فى الحالة المحمية protected mode فإنه فى هذه الحالة يتعامل مع 64 تريليون بايت (1تريليون بايت=1024 جيجابايت) من الذاكرة التخيلية وذلك باستخدام وحدة إدارة الذاكرة MMU .

17-4-1 التركيب الهيكلى Architecture للمعالج 80386

شكل (17-4) يبين الشكل الخارجى لشريحة هذا المعالج وطريقته الجديدة فى ترتيب أطرافه ، حيث يخرج من هذه الشريحة 132 طرفا مرتبة فى صورة شبكة Grid تعرف كل نقطة فيها برقم الصف متقاطعا مع رمز العمود الذى تقع فيه ، فنقول مثلا الطرف 13j هو الطرف Vss وهكذا .
المعالج i386 (اختصار 80386) نزل فى إصدارين أو صورتين ، الإصدار الأول هو المعالج i386DX وهو الصورة الكاملة لهذا المعالج والتى نحن بصدد دراستها هنا . الإصدار الثانى هو المعالج i386SX الذى يختلف عن الإصدار DX فى أن مسار البيانات له يتكون من 16 بت بدلا من 32 وذلك حتى يتوافق خارجيا مع المعالج 80286 وهذا هو الاختلاف الوحيد بينهما .

17-4-2 تنظيم الذاكرة للمعالج 80386

عندما كان مسار البيانات 8 بت كما فى المعالجات 8085 أو Z80 كانت الذاكرة تنظم فى صورة بنك bank واحد ، عرض هذا البنك هو 8 بت (نفس عرض مسار البيانات) . عندما تطور مسار البيانات إلى 16 بت أصبحت الذاكرة تنظم فى صورة بنكين كل منهما 8 بت بحيث يكون البنك الأول للبايتات الزوجية والثانى للبايتات الفردية ، وكان الخط A0 يستخدم لتنشيط البنك الزوجى أو النصف الأدنى فى حالة التعامل مع هذا البنك فقط ، والخط $\overline{BHE}$ يستخدم لتنشيط البنك الفردى أو النصف العلوى فى حالة التعامل معه فقط ، أما فى حالة التعامل

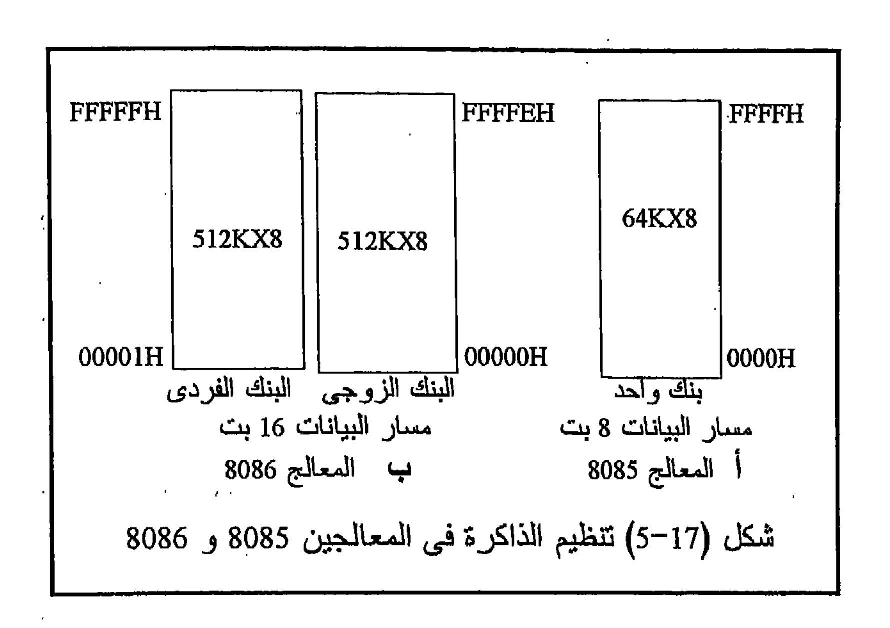
على مستوى 16 بت فإن كل من البنكين يتم تنشيطهما فـــى نفـس الوقــت مــن الخطين A0 و $\overline{BHE}$ حتى يمكن إرسال 16 بت (كلمة word) مرة واحدة ، ولقــد رأينا ذلك فى أثناء دراستنا للمعالج 8086 . شكل (17-5) يبين طريقـــة تنظيــم الذاكرة فى المعالجين 8085 و 8086 .

	P	N	M	L	K	J	H	G	F	E	D	C	B	A	
1	A30	A27	A26	A23	A21	A20	A17	A16	A15	A14	A11	A8	Vss	Vcc	1
2	Vcc	A31	A29	A24	A22	Vss	A18	Vcc	Vss	A13	A10	A7	A5	Vss	2
3	D30	Vss	Vcc	A26	A25	Vss	A19	Vcc	Vss	A12	A9	A6	A4	A3	3
4	D29	Vcc	Vss									A2	NC	NC	4
5	D26	D27	D31									Vcc	Vss	Vcc	5
6	Vss	D25	D28									NC	NC	Vss	6
7	D24	Vcc	Vcc									NC	INTR	Vcc	7
8	Vcc	D23	Vss									PEREQ	NMI	ERROR	8
9	D22	D21	D20									RESET	BUSY	Vss	9
10	D19	D17	Vss									LOCK	W/R	Vcc	10
11	D18	D16	D15									Vss	Vss	D/C	11
12	D14	D12	D10	Vcc	D7	Vss	D0	Vcc	CLK	BEQ	Vcc	Vcc	NC	M/IO	12
13	D13	D11	Vcc	D8	D5	Vss	D1	RDY	NC	NC	NA	BE1	BE2	BE3	13
14	Vss	D9	HLDA	D6	D4	D3	D2	Vcc	Vss	ADS	HOLD	BS16	Vss	Vcc	14
	P	N	M	L	K	J	H	G	F	E	D	C	B	A	

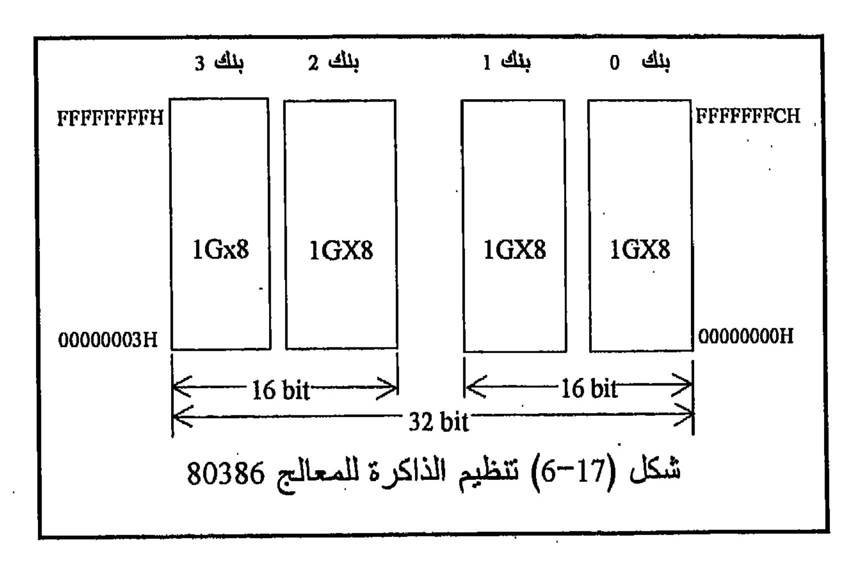
شكل (17-4) الرسم الطرفى للمعالج 80386

مسار البيانات فى المعالج i386 مكون من 32 بت ، أى أنه سيتعامل مع ذاكـــرة مقدارها 4 جيجابايت ستقسم كما فى شكل (17-6) فى صورة 4 بنكات كل بنـــك سيكون له خط تنشيط منفصل وهى الخطوط $\overline{BE0}$ إلى $\overline{BE3}$ بحيث أنـــه عندمـــا يتعامل على مستوى بايت واحدة فإنه يتم تنشيط البنك المطلوب بخـــط التنشـــيط المناسب له ، وعندما يتعامل على مستوى 16 بت فإنه ينشط إما الخطيـن $\overline{BE0}$ و

$\overline{BE1}$ فى نفس الوقت فى حالة التعامل مع الكلمة الأولــى ، أو الخطيــن $\overline{BE2}$ و $\overline{BE3}$ فى نفس الوقت فى حالة التعامل مع الكلمة الثانية (العليا) . أما فــى حالــة التعامل على مستوى 32 بت (4 بايت) ففى هذه الحالة تنشط كل الخطــوط $\overline{BE0}$ إلى $\overline{BE3}$ فى نفس الوقت .



شكل (17-5) تنظيم الذاكرة فى المعالجين 8085 و 8086



شكل (17-6) تنظيم الذاكرة للمعالج 80386

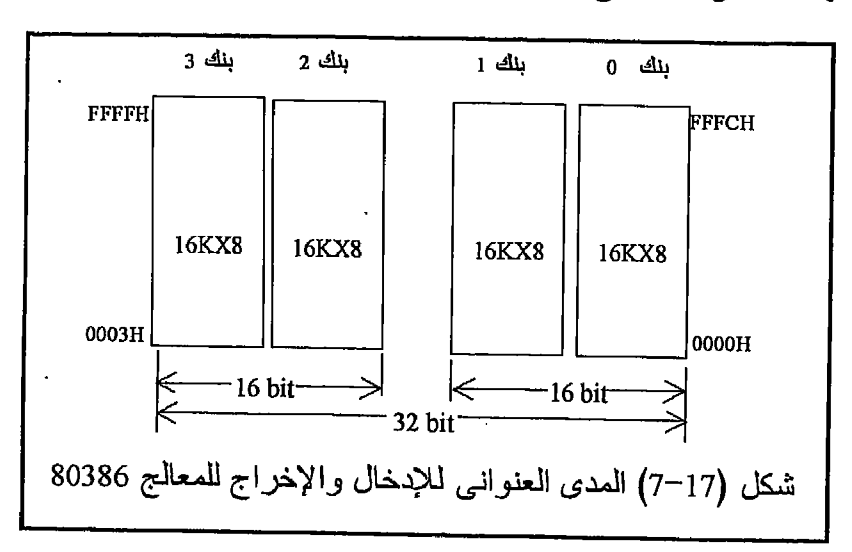
عند إعادة وضع reset المعالج i386 فإنه يذهب إلى العنوان FFFFFFF0H حيث يبدأ التنفيذ من هناك .

17-4-3 نظام الإدخال والإخراج فى المعالج 80386

المعالج i386 مثل ما سبقه من المعالجات يستطيع التعامل مع عدد من بوابات الإدخال أو الإخراج يصل إلى 64 كيلو ، أى أن المدى العنوانى للإدخال والإخراج هو من صفر إلى FFFFH . الجديد هنا سيكون فى طريقة تنظيم هذه العناوين فى صورة بنكات نتيجة كون مسار البيانات أصبح 32 بت . نتيجة لذلك سيقسم هذا المدى العنوانى إلى 4 بنكات كما فى شكل (17-7) حيث ستستخدم خطوط التنشيط $\overline{BE0}$ إلى $\overline{BE3}$ لتنشيط البنك المناسب للتعامل معه سواء كان التعامل فى صورة 8 أو 16 أو 32 بت .

17-4-4 أطراف المعالج 80386

1. الأطراف A0 إلى A31 تمثل مسار العناوين ، وتستخدم لعنونة 4 جيجابايت كما ذكرنا ، الجديد هنا أن خطوط العناوين والبيانات ليست ممزوجة زمنيا مع بعضها كما كان الحال فى المعالجات السابقة .



شكل (17-7) المدى العنوانى للإدخال والإخراج للمعالج 80386

2. الأطراف D0 إلى D31 تمثل مسار البيانات .
3. الأطراف $\overline{BE0}$ إلى $\overline{BE3}$ هى خطوط تنشيط البنوك المختلفة فى الذاكرة والإدخال والإخراج على حسب نظام التعامل 8 أو 16 أو 32 بت .
4. الطرف $M/\overline{IO}$ طرف خرج يبين إذا كان العنوان الموجود على مسار العناوين يمثل ذاكرة (حيث يكون هذا الطرف واحد) أم عنوان لبوابة إدخال أو إخراج (حيث يكون هذا الطرف صفر) .

5. الطرف $W/\overline{R}$ طرف خرج يبين إذا كان التعامل الحـــالى ســيكون بغــرض القراءة حيث يكون هذا الطرف صفرا أم الكتابة حيث يكون هذا الطرف واحــد . لاحظ أنه فى كل المعالجات السابقة كان هناك خطان أحدهما للقراءة $\overline{RD}$ والآخر للكتابة $\overline{WR}$.

6. الطرف $\overline{ADS}$ طرف خرج يحمل إشارة تبين إذا كانت الإشارة الموجودة علـى مسار العناوين تمثل عنوان محقق للذاكرة أو لبوابة إدخال أو إخــراج ADdress Status . هذا الخط يستخدم فى العادة مع الخط $W/\overline{R}$ للحصول على الإشـــارات $\overline{MEMR}$ و $\overline{MEMW}$.

7. الطرف RESET ، طرف دخل عندما يكون واحد يسبب إعادة وضــــع reset للمعالج حيث يذهب المعالج للعنوان FFFFFFF0H ويبدأ التنفيذ من هناك .

8. الطرف CLK2 ، طرف دخل يحمل نبضات الساعة Clock للمعالج . تـــردد هذه النبضات يجب أن يكون ضعف التردد المطلوب للمعالج لأنه يتم قسمة هـــذا التردد على 2 قبل استخدامه داخل المعالج .

9. الطرف $\overline{READY}$ ، طرف دخل يستخدم لإدخال دورات انتظار على المعــالج حينما يكون نشط (0) .

10. الطرف $\overline{LOCK}$ يستخدم لمنع أى جهاز خارجى أو معالج مساعد مثل المساعد الحسابى i387 من الحصول على المسارات .

11. الطرف $D/\overline{C}$ ، طرف خرج يعنى Data/Control ويبين إذا كـــانت الإشـــارة الموجودة على مسار البيانات تمثل بيانات أم إشارة تحكم يخرجها المعـــالج عنــد تنفيذ الأمــر HALT أو أنـــه يرســل إشـــارة اعـــتراف بالمقاطعـــة Interrupt Acknowledge .

12. الطرف $\overline{BS16}$ ، طرف دخل يستخدم لتغيير نظام العمل على مسار البيانــــات بجعله 16 بت بدلا من 32 بت . حينما يكون هذا الخط صفرا يتعـــامل المعـــالج على أساس أن مسار البيانات 16 بت ، وحينما يكون واحد يعتبر مسار البيانـــات 32 بت .

13. الطرف $\overline{NA}$ ، ويعنى Next Address أو العنوان التالى ، وهو طــرف دخــل يستخدم لجعل المعالج يخرج العنوان التالى فى أثناء تنفيذ الأمر الحـــالى حيــث تستخدم هذه الطريقة لإسراع عملية الاتصال بالذاكرة .

14. الطرفان HOLD و HLDA مثل نظيريهما فى المعالجات السابقة .

15. الطرف $\overline{PEREQ}$ طرف دخل يسمح للمعالج الحسابى i387 بطلب بيانات مــن المعالج i386 .

16. الطرف $\overline{BUSY}$ ، طرف دخل يستخدم حينما يكون صفرا لإخبـــار المعـــالج i386 بأن المساعد الحسابى i387 مشغول وليس على استعداد لاســـتقبال أوامـــر أخرى الآن .

17. الطرف $\overline{\text{ERROR}}$ ، طرف دخل يستخدمه المعالج الحسابى لإخبـــار المعـــالج i386 بأن هناك خطأ قد حدث .
18. الطرف INTR ، طرف دخل يستخدم لطلب المقاطعة .
19. الطرف NMI ، طرف دخل يستخدم لطلب المقاطعة الغير محجوبة .

5-4-17 مسجلات المعالج 80386

شكل (17-8) يبين التركيب الهيكلى والمسجلات الموجودة داخل المعـــالج i386 . نلاحظ من هذا الشكل أن نفس عدد المسجلات مازال موجودا فى هـــذا المعـــالج وهذه المسجلات مازالت تؤدى نفس الدور . الجديد هنا هو أن المســـجلات فـــى المعالج i386 تستطيع التعامل مع بيانات مقدارها 8 أو 16 أو 32 بـــت . حينمـــا نريد التعامل مع هذه المسجلات على أساس 32 بـــت فإننـــا نضـــع الحـــرف E (اختصار لكلمة ممتد Extended) أمام المسجل المراد التعامل معه كما فى الأمـــر التالى :

EAX		AH	AL	AX
EBX		BH	BL	BX
ECX		CH	CL	CX
EDX		DH	DL	DX
ESP		SP		
EDP		DP		
EDI		DI		
ESI		SI		

المسجلات عامة الأغراض

CS
DS
ES
SS
FS
GS

مسجلات التجزىء

EIP		IP
EFLAGS		FLAGS

شكل (17-8) مسجلات المعالج 80386

MOV EAX,FF340056H

حيث EAX معناها مسجل التراكم الممتد ، وهكذا باقى المسجلات العامة .
مسجلات التجزيء GS, FS, ES, SS, DS, CS تلعب نفس الدور الذى كانت تلعبه مع المعالج 8086 فى الحالة الحقيقية real mode ، وتلعب مع المسجلين GS, FS أدوارا إضافية فى الحالة التخيلية virtual mode .
هناك أيضا المعالج EIP الذى يمثل مؤشر الأوامر الممتد والذى يستطيع التعامل مع 32 بت ، كذلك مسجل الأعلام هنا أصبح ممتدا أيضا حيث أصبح اسمه EFLAGS .
قبل أن نترك هذا المعالج نؤكد أن جميع أوامر المعالج 8086 مازالت محققة ويمكن استخدامها بالكامل وبدون أى تعديل مع المعالج i386 ، الفرق هو أن المعالج i386 يستطيع التعامل مع بيانات من 8 أو 16 أو 32 بت ، فكل الأوامر التالية صحيحة :

MOV AL, 55H
MOV AX, 5544H
MOV EAX, 55443322H

5-17 الذاكرة المخبأة Cache Memory

مع زيادة نبضات الساعة clock للمعالج (33ميجاهرتز للمعالج i386) أصبح زمن تنفيذ أى أمر صغيرا جدا بحيث أصبح أقل من زمن الاتصال بالذاكرة مما سيتسبب فى وجود فترات انتظار عند تنفيذ أى أمر يتعامل مع الذاكرة وبالتالى تقليل سرعة المعالج . زمن الاتصال بالذاكرة هو الزمن اللازم لقراءة أو كتابة وحدة بيانات فى الذاكرة ، وهذا الزمن يكون عادة فى حدود 50 نانوثانية بالنسبة للذاكرة الديناميكية . للتغلب على هذه المشكلة تم استخدام أسلوب الذاكرة المخبأة cache ، وهى عبارة عن كمية من الذاكرة السريعة جدا التى تتميز بصغر زمن الاتصال بها والتى تصنع خصيصا ، ولذلك فإنها مرتفعة الثمن جدا . كمية هذه الذاكرة تبدأ من 8 كيلوبايت وتصل إلى 512 كيلوبايت وكانت هذه الذاكرة توصل خارج المعالج ، أما الآن فإنها توصل داخل شريحة المعالج نفسه كما سنرى عند عرضنا للمعالجات الحديثة مثل عائلة بنتيم Pentium .
من المعروف أن التعامل مع بايتات الذاكرة يكون فى الغالب من أماكن متتابعة فى الذاكرة ، بمعنى أنه عند القراءة أو الكتابة من بايت معينة فإنه فى الغالب يكون التعامل التالى مع الذاكرة من البايت التالية للبايت السابقة . لذلك عندما يقرأ المعالج بايت معينة من الذاكرة فإنه يحضر هذه البايت وكمية من البايتات

التالية لها ويضعها فى الذاكرة المخبأة على أمل أن يكون التعامل التالى مع الذاكرة المخبأة وليس مع الذاكرة الأساسية . ولذلك فإن المعالج عندما يقرأ بايت من الذاكرة فإنه يبحث عن هذه البايت أولا فى الذاكرة المخبأة ، فإن وجدها فإنه سيقرأها بأقل زمن اتصال ، وإذا لم يجدها فإنه يحضرها من الذاكرة الأساسية وفى نفس الوقت يضعها أيضا فى الذاكرة المخبأة مع محاولة ملأ الذاكرة المخبأة بالبيانات التالية لهذه البايت . عملية ملأ الذاكرة المخبأة تتم عادة فى أثناء فترات انتظار المعالج . بذلك نضمن أن البايت التى من المحتمل أن يتم قراءتها فى المرة القادمة ستكون موجودة فى الذاكرة المخبأة . عملية الكتابة فى الذاكرة تكون بنفس الطريقة ، فإن كانت المعلومة المراد إرسالها إلى الذاكرة موجودة فى الذاكرة المخبأة فإنه يتم نقلها بأقل زمن اتصال ممكن ، وإذا لم تكن موجودة يتم تسجيلها والمعلومات التالية لها فى الذاكرة المخبأة أولا ثم ترسل إلى الذاكرة الأساسية ، بذلك نضمن أن المعلومة التى ستكتب فى الذاكرة فى المرة القادمة ستكون موجودة غالبا فى الذاكرة المخبأة . أى أن عمليات الكتابة أو القراءة من أو إلى الذاكرة الأساسية تكون من خلال الذاكرة المخبأة ، ودون تدخل من المستخدم ، وهذا هو السبب فى تسميتها بالذاكرة المخبأة cache لأنها تكون مخبأة عن المستخدم وليس له دخل فى التعامل معها أو إدارتها . هذه العملية ثبت أنها تزيد جدا من سرعة التعامل مع الذاكرة اعتمادا على حقيقة أن البيانات التى يتم التعامل معها فى أى وقت سيتم التعامل مع المعلومة التالية لها فى المرة القادمة .

17-6 المعالج 80486

المعالج 80486 هو معالج عالى التكامل حيث يحتوى بداخله المعالج الحسابى الخاص به 80487 بالإضافة إلى وحدة إدارة الذاكرة وكمية من الذاكرة المخبأة cache memory تبلغ 8 كيلوبايت ، كل ذلك مجمع على نفس شريحة المعالج . لك أن تتخيل مدى كثافة المكونات فى هذه الشريحة إذا علمت أنها تحتوى على أكثر من مليون ترانزستور . هذا المعالج يستطيع تنفيذ كل أوامر المعالجات السابقة له من عائلته بدون أى تعديل . بالطبع فإنه لابد أن يحتوى على بعض الأوامر الإضافية نتيجة الإضافات التى تضاف عليه . هذا المعالج يستخدم فكرة مجموعة الأوامر المخفضة ، Reduced Instruction Set Computer, RISC ، والتى ساعدت مع عوامل أخرى فى تخفيض الزمن اللازم لتنفيذ الكثير من الأوامر إلى نبضة تزامن واحدة . هذا بالإضافة إلى الذاكرة المخبأة cache memory وسرعة نبضات التزامن العالية التى أمكن الوصول إليها فى ذلك الوقت والتى بلغت 33 أو 66 ميجاهرتز ، كل ذلك جعل سرعة تنفيذ البرمجيات بهذا المعالج تبلغ أضعاف سرعتها باستخدام المعالج i386 .

يوجد المعالج i486 في صورة شريحة شبكية Grid ذات 168 طرفا . مسار العناوين لهذا المعالج يتكون من 32 بت ، وكذلك مسار البيانات . بالنسبة للتركيب الهيكلى لهذا المعالج فإن مجموعة المسجلات الموجودة فيه هى نفسها مجموعة المسجلات الموجودة فى سابقه المعالج i386 . نخلص من ذلك أن المعالج i486 هو نفسه المعالج i386 مضافا إليه المساعد الحسابى i487 وذاكرة مخبأة مقدارها 8 كيلوبايت .

إننا هنا لن ندخل فى تفاصيل أخرى عن هذا المعالج ولا المعالجات التالية ، ولكننا سنكتفى فقط بذكر الجديد أو الإضافة التى قدمتها هذه المعالجات وسنترك الأمر لمن يريد الاستزادة أن يرجع إلى المراجع الموجودة فى نهاية الكتاب أو الكتالوجات الخاصة بالمعالج الذى يريد دراسته بالتفصيل .

17-7 انسيابية الأوامر Instruction Pipelining

سنقدم هنا فكرة جديدة تزيد سرعة تنفيذ الأوامر بدرجة كبيرة جدا تصل إلى خمس مرات على الأقل . تخيل أن أى أمر يحتاج إلى خمس نبضات تزامن حتى يتم تنفيذه ، بحيث تتم عملية التنفيذ فى خلال الخمس نبضات بالخطوات التالية :

1- إحضار الأمر Fetch Instruction, FI
2- تشفير الأمر Decode Instruction, DI
3- إحضار المعاملات Fetch Operand, FO
4- تنفيذ الأمر Execute Instruction, EX
5- تخزين النتيجة Write Result, WR

هذه الخطوات الخمس يمكن تنفيذها بالتتابع على أى أمر ، وفى هذه الحالة فإننا سنحتاج إلى خمس نبضات تزامن لكى تتم عملية إحضار وتنفيذ أى أمر . يمكن إسراع هذه العملية باستخدام فكرة انسياب الأوامر كما هى موضحة فى شكل (17-9) . نلاحظ من هذا الشكل أن كل أمر تم تقسيمه إلى خمس مراحل بحيث عندما يكون المعالج مشغولا فى تنفيذ مرحلة معينة لأمر معين فإن الأمر التالى يتم تنفيذه أيضا فى نفس الوقت ولكن فى مرحلة أخرى من مراحل التنفيذ . فمثلا عندما يكون الأمر رقم I فى مرحلة التخزين WR فإن الأمر I+1 يكون فى مرحلة التنفيذ EX ، ويكون الأمر I+2 فى مرحلة إحضار المعاملات ، والأمر I+3 فى مرحلة تشفير الأمر ، والأمر I+4 فى مرحلة إحضار الأمر ، وهكذا . أى أنه يكون هناك دائما خمسة أوامر موجودة داخل وحدة التنفيذ كل أمر منها يتم تنفيذ مرحلة معينة منه على حسب موقعة فى تتابع الأوامر داخل الوحدة . نلاحظ من هذا الشكل أننا سنحصل من وحدة التنفيذ على أمر وقد تم تنفيذه فى

نهاية كل نبضة تزامن (أمر لكل نبضة) . أى أن سرعة تنفيذ الأوامر قد زادت بمقدار خمس مرات ويمكن زيادتها أكثر من ذلك بزيادة عدد مراحل تنفيذ الأوامر . أى أن الأوامر تنساب فى مراحل التنفيذ فيما يشبه الأنبوبة أو خط الإنتاج وكل أمر يوجد فى مرحلة تنفيذ معينة ، ولذلك سميت هذه الطريقة بالانسيابية أو Pipelining.

رقم الأمر	نبضات التزامن							
	1	2	3	4	5	6	7	8
I	FI	DI	FO	EX	WR			
I+1		FI	DI	FO	EX	WR		
I+2			FI	DI	FO	EX	WR	
I+3				FI	DI	FO	EX	WR
I+4					FI	DI	FO	EX
I+5						FI	DI	FO

شكل (17-9) الانسيابية **Pipelining**

من الواضح أنه لكى نستفيد من فكرة الانسيابية فإن جميع الأوامر لابد أن يكون لها نفس الطول أو نفس عدد المراحل ، وكل مرحلة لابد أن تنفذ فى نبضة تزامن واحدة ، فهل هذا محقق فى أوامر المعالجات التى تمت دراستها حتى الآن ؟ بالطبع الإجابة هى لا ، فإن أوامر جميع المعالجات التى درسناها حتى الآن لها أطوال مختلفة وتنفذ فى أعداد مختلفة من نبضات الساعة . وهذا يسوقنا إلى تقسيم المعالجات إلى نوعين من حيث مجموعة أوامر كل منها .

النوع الأول يسمى المعالجات ذات مجموعة الأوامر المركبة ،
Complex Instruction Set Computers, CISC

النوع الثانى يسمى المعالجات ذات مجموعة الأوامر المخفضة ،
. Reduced Instruction Set Computers, RISC

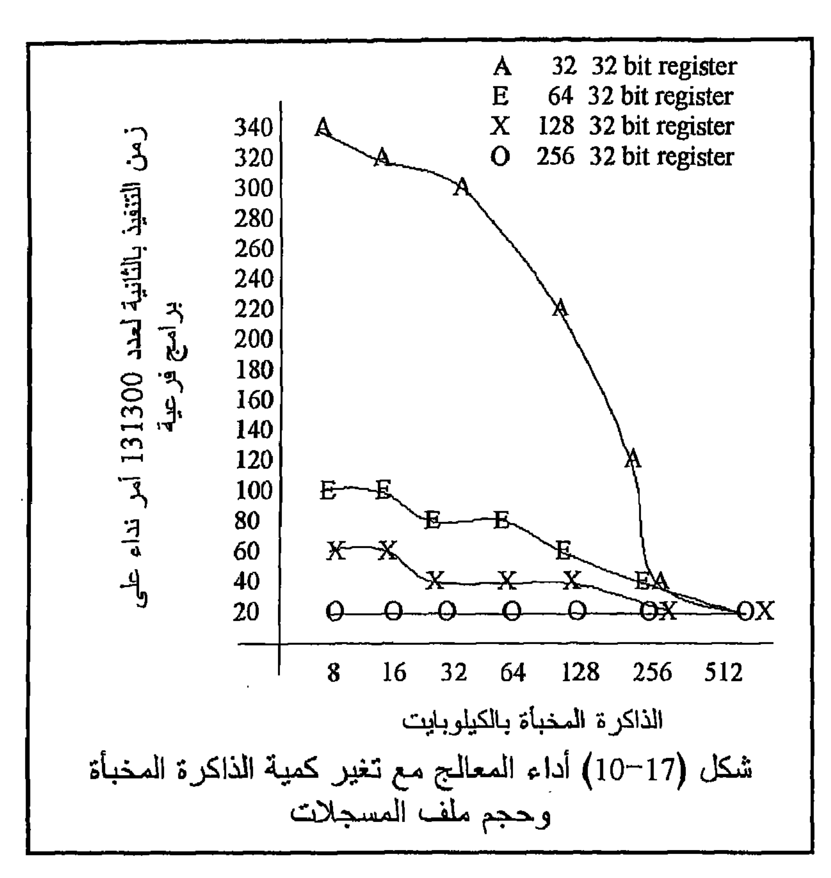
فى النوع الأول من المعالجات ، CISC ، تكون كمية الأوامر كبيرة جدا ، 300 أمر على الأقل ويكون معظم هذه الأوامر أوامر مركبة . ولذلك فإن مشفر الأوامر Instruction decoder يكون معقدا جدا مما يتسبب أن الإشارة تأخذ وقتا كبيرا فى تخلل دائرة المشفر ، ولذلك فإن وحدة التحكم فى هذه المعالجات تكون معقدة . أيضا فإن الكثير من الأوامر المركبة يتم تنفيذها بطريقة البرمجيات الصغيرة Microprograms حيث ينفذ أمر الضرب مثلا بتنفيذ برمجية صغيرة

تنفذ عملية الضرب فى صورة مجموعة من أوامر الجمع المتكرر ، وهذا بالطبع يستهلك الكثير من الوقت . كذلك فإنه نتيجة اختلاف أطوال الأوامر فى هذا النوع من المعالجات فإنه يكون من الصعب استخدام فكرة الانسيابية Pipelining . نتيجة لذلك ظهر التفكير فى النوع الثانى من المعالجات RISC حيث يكون كل شيء هنا مخفض ، عدد الأوامر تم تخفيضه حتى أقل من 128 أمر ، وكذلك طرق عنونة الذاكرة Memory addressing تم تخفيضها حيث أن التعامل مع الذاكرة يستهلك الكثير من الوقت .

مع بساطة عدد الأوامر فإن مشفر الأوامر ، وكذلك وحدة التحكم سيكونان أكثر بساطة مما سيوفر الكثير من وقت التنفيذ . كذلك فإن تخفيض عدد الأوامر سيقتصر على الأوامر ذات الأطوال الواحدة بقدر الإمكان والتى يمكن تنفيذها فى نفس عدد المراحل مما سيسهل استخدام طريقة الانسيابية Pipelining فى تنفيذ هذه الأوامر . مع تخفيض عدد الأوامر وبساطتها فى المعالجات RISC أمكن الاستغناء وبدرجة كبيرة على استخدام البرمجيات الصغيرة فى تنفيذ الأوامر حيث أمكن استخدام دوائر مبنية Hardware لتنفيذ هذه الأوامر مما وفر الكثير من وقت التنفيذ أيضا . كما رأينا فإن أنظمة RISC تتمتع بالكثير من المميزات مما جعل معظم المعالجات ابتداء من المعالج 80486 تقريبا يأخذ بهذه الفكرة وينفذها ، حيث أصبحت كل الأوامر تقريبا تنفذ فى نبضة تزامن واحدة .

فى أثناء تنفيذ بعض الأوامر فى أى برنامج تنتج هناك بعض النتائج المرحلية التى يحتاجها البرنامج بعد قليل ، ولكن بما أن عدد المسجلات داخل المعالج يكون محدودا فإن المعالج يضطر لإرسال هذه النتائج المرحلية إلى الذاكرة حيث يتم استدعاؤها مرة ثانية عند الحاجة إليها ، وهذا بالطبع يستهلك الكثير من الوقت . هذه المشكلة يمكن التغلب عليها بدرجة كبيرة بزيادة عدد المسجلات ذات الأغراض العامة داخل المعالج بحيث يمكنها أن تستوعب هذه النتائج المرحلية . معظم معالجات RISC تحتوى على عدد كبير من المسجلات تصل إلى 32 مسجلا وتسمى هذه المجموعة بملف المسجلات Register file . هنا يمكن أن نسأل السؤال التالى : هل يمكن الاستغناء عن الذاكرة المخبأة Cache memory باستخدام ملف مسجلات مع زيادة عدد المسجلات فيه ؟ الإجابة على هذا السؤال هى لا . إن زيادة عدد المسجلات بدرجة كبيرة يجعل من الصعب عنونتها ويكون التعامل مع الذاكرة المخبأة فى هذه الحالة أسهل . إن الحد الفاصل بين عدد المسجلات الكبير والصغير غير واضح تماما ولكن ثبت بالتجربة أن 32 أو 64 مسجلا بالإضافة إلى كمية من الذاكرة المخبأة يكون لها أفضل تأثير على سرعة أداء المعالج . شكل (17-10) [Tabak 1995] يبين علاقة بين كمية المسجلات المستخدمة مع كمية الذاكرة المخبأة على أداء المعالج من حيث سرعة تنفيذ مجموعة من أوامر النداء على البرامج الفرعية . من هذا

الشكل نلاحظ كيف أن زيادة عدد المسجلات من 32 إلى 64 مسجلا كان له تأثيرا كبيرا على زيادة السرعة ، ولكن معدل هذا التحسين كان قليلا جدا مع زيادة عدد المسجلات عن 64 مسجلا . ولذلك فإنه ثبت بالتجربة وكما هو واضح من هذا الشكل أن 64 مسجلا مع 256 كيلوبايت من الذاكرة المخبأة يعطى أحسن أداء للمعالج .



شكل (17-10) أداء المعالج مع تغير كمية الذاكرة المخبأة وحجم ملف المسجلات

من الأسباب المهمة التى تجعل من الصعب الاستغناء عن المسجلات العامة بالذاكرة المخبأة أن المسجلات العامة يمكن استخدامها بواسطة المبرمج وتكون دائما تحت تصرفه ، بينما فى حالة الذاكرة المخبأة فإن المبرمج لا يملك أى سيطرة عليها ولا يستطيع التعامل معها على الإطلاق والمتحكم فى تشغيلها فقط هو وحدة التحكم بالمعالج . لذلك لابد من وجود ملف مسجلات مع كمية من الذاكرة المخبأة للحصول على أحسن أداء للمعالج .

وعلى ذلك يمكن تلخيص مجموعة الخواص المميزة لأى حاسب RISC فيما يلى:

1. جميع الأوامر (أو 80% على الأقل) تنفذ فى نبضة تزامن واحدة .

2. جميع الأوامر لها نفس الطول (عدد البايتات لكل أمر) ، وفى الغالب يتكـــون كل أمر من 4 بايت (32 بت) .
3. عدد مخفض من الأوامر لا يتعدى 128 أمر .
4. عدد مخفض من طرق التعامل مع الذاكرة Addressing modes لا يزيد عـــن 4 طرق .
5. كل الأوامر فيما عدا القليل منها يتعامل مع المسجلات فقط .
6. وحدة التحكم تكون مصممة باستخدام الدوائـــر المبنيـــة Hardwired وليـــس باستخدام البرمجيات الصغيرة Microprograms .
7. عدد كبير من المسجلات العامة الأغراض (32 مسجل على الأقل) فى ملـــف مسجلات .

ومن مميزات معالجات RISC ما يلى :

1. البناء باستخدام تكنولوجيا التجميع عالى الكثافة جدا VLSI نتيجة لما يلى :

- نتيجة تبسيط وحدة التحكم تقل مساحته بدرجة كبـــيرة ، ويكفـــى أن نعلـــم أن وحدة التحكم فى معالجات CISC تشغل تقريبا 50% من مساحة الشريحة بينمـــا تشغل فقط حوالى 10% من مساحة الشريحة فى حالة المعالجات RISC .
- نتيجة صغر مساحة وحدة التحكم على الشريحة يمكن إضافة ملف مســـجلات يحتوى عددا أكبر من المسجلات .

2. زيادة سرعة الحساب نتيجة لما يلى :

- إمكانية استخدام الانسيابية Pipelining فى تنفيذ الأوامر فإن المعـــالج يكـــون مشغولا دائما مما يزيد من سرعته .
- تصميم وحدة التحكم باستخدام الدوائـــر Hardwired بـــدلا مـــن البرمجيـــات الصغيرة Microprograms .
- وجود ملف المسجلات يقلل التعامل مع الذاكرة .

3. بساطة الأوامر المستخدمة وتخفيض عددها ، وبساطة تصميم وحدة التحكـــم بالطبع سينعكس على تكلفة تصميم شريحة المعالج مما ســـينعكس علـــى ســـعر المعالج .

من عيوب المعالجات RISC أن تخفيض عدد الأوامر سيضطر مصمموا الـبرامج إلى استخدام عدد أكثر من الأوامر لتنفيذ نفس البرنامج مما سيؤدى إلى كبر حجـم البرنامج وكبر حجم الذاكرة التى يشغلها . لذلك فإنه مـــن المتوقـــع أن برامـــج معالجات RISC تكون أطول بحوالى 30% من برامج معالجات CISC .

8-17 سلسلة معالجات بنتيم
Pentium Processors

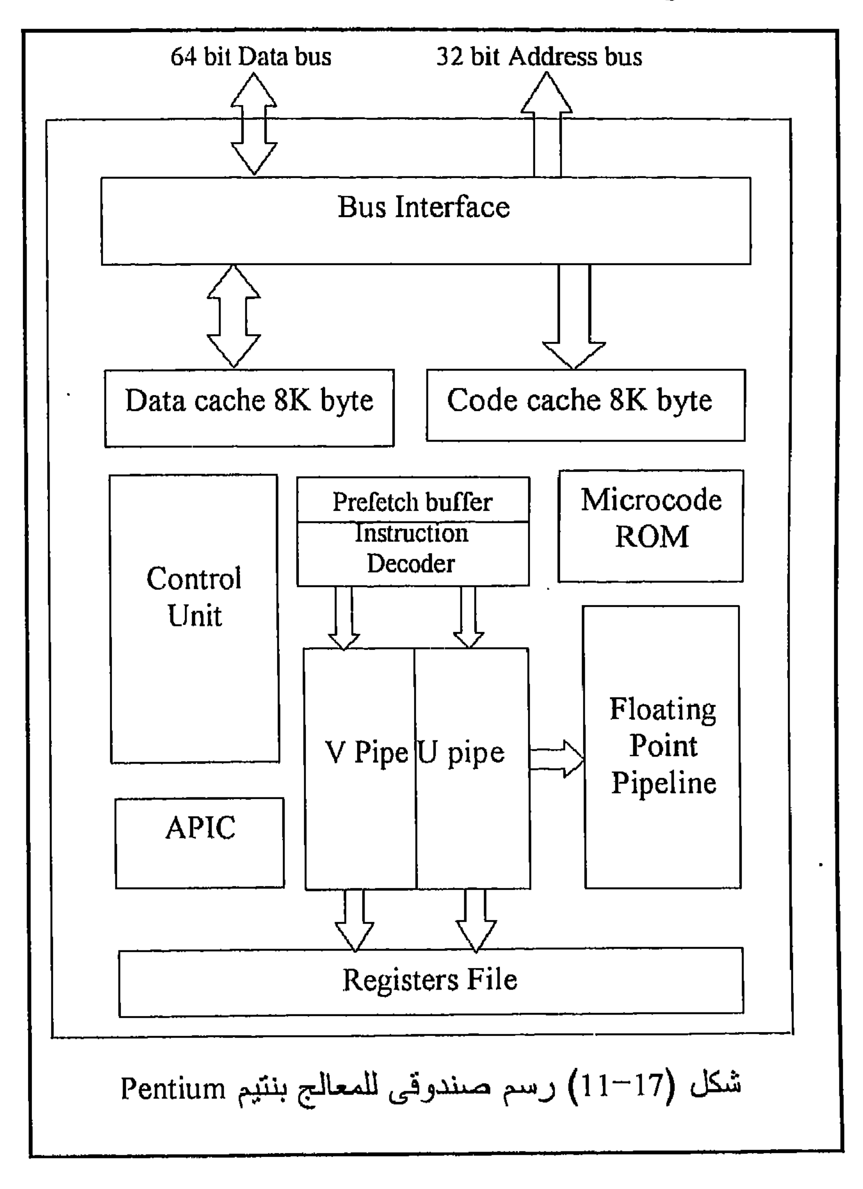
بعد المعالج 80486 وفى بداية التسعينات ، 1993 ، ظهرت سلسلة المعالجات بنتيم pentium ، وسنطلق عليها سلسلة لأنها خضعت لتطورات سريعة جدا ومتلاحقة . هذه المعالجات تستخدم طريقة القوائم المخفضة RISC واثنان من الانسيابات Pipelines لزيادة سرعة تنفيذ الأوامر حيث وصلت سرعة تنفيذ الأوامر فيه إلى 330 MIPS (Million Instruction Per Second) للمعالج الذى يعمل بنبضات ساعة مقدارها 200 ميجاهرتز بدلا من 54 MIPS للمعالج i486DX2 الذى يعمل بنبضات ساعة مقدارها 66 ميجاهرتز . من أهم خواص هذا المعالج ما يلى :

1. اثنان من الانسيابات Pipelines واحدة لتنفيذ الأوامر التى تتعامل مع البيانات الصحيحة Integer Pipelines والأخرى لتنفيذ الأوامر التى تتعامل مع البيانات الحقيقية Floating Point Instructions .
2. خاصية توقع أوامر التفريع مثل القفز والنداءات على البرامج الفرعية ، والتى يكون لها دخل كبير فى إسراع التعامل مع الذاكرة المخبأة .
3. ذاكرة مخبأة خاصة بالتعامل مع البيانات ، وأخرى خاصة بالتعامل مع الأوامر .
4. مسار بيانات خارجى 64 بت .
5. حالة تشغيل جديدة وهى حالة توفير القدرة Power saving mode .

شكل (17-11) يبين رسما صندوقيا لهذا المعالج حيث نلاحظ وجود وحدتى الذاكرة المخبأة وحدتى الانسياب المنفصلتين حيث بهذه الطريقة يمكن تنفيذ أمرين فى نفس الوقت على التوازى . أحد إصدارات هذا المعالج يوجد فى شريحة ذات 296 طرفا فى الشكل الشبكى وتسحب تيارا مقداره 4 أمبير للمعالج 133 ميجاهرتز . هذا المعالج له 53 طرفا كلها تمثل القدرة Vcc ، و 53 طرفا تمثل الأرضى . هذا العدد الكبير من الأطراف الخاصة بالقدرة يستخدم لتخفيض الحرارة المنبعثة . لاحظ ازدواجية وحدة الانسياب ووجود وحدة جديدة APIC وهى وحدة تحكم فى المقاطعة قابلة للبرمجة (Advanced Programmable Interrupt Controller) . وحدة التحكم الموجودة فى المعالج بنتيم تتحكم فى دائرتى الانسياب حيث فى الأحوال العادية يستطيع المعالج تنفيذ أمرين فى نفس الوقت كما ذكرنا . يستطيع المعالج بنتيم تنفيذ عمليات البيانات الحقيقية Floating

Point أسرع من المعالج i486 حوالى 10 مرات نتيجة اســتخدام دوائــر مبنيــة Hardware لتنفيذ عمليات الضرب و القسمة بدلا من البرمجيات الصغيرة .

يحتوى المعالج بنتيم على بعض الأوامر المركبة التى تعمل باستخدام البرمجيــات الصغيرة وبالذات الأوامر المنقولة من المعالجات السابقة ، ولذلك فإن هذا المعالج متوافق تماما مع كل ما سبقه من معالجات بحيث أن كل أوامر المعالجات الســابقة يمكن تنفيذها عليه مع الاستفادة بوحدتى الانسياب الموجودة فيه .



شكل (17-11) رسم صندوقى للمعالج بنتيم Pentium

المدى العنوانى للإدخال والإخراج I/O Address Space للمعالج بنتيم يبلغ 64 كيلوبايت للبوابات ذات 8 بت ، أو 32 كيلوبايت للبوابات ذات 16 بت أو 16 كيلوبايت للبوابات ذات 32 بت حيث يمكن للمعالج التعامل مع كل هذه الأنواع أو مع بعضها .

الذاكرة المخبأة تكون غالية الثمن كما ذكرنا لأنها تتميز بصغر زمن الاتصال بها وتنقسم هذه الذاكرة إلى مستويين من حيث اتصالها بالمعالج . فهناك الذاكرة المخبأة ذات المستوى الأول Level 1 والتى يرمز لها بالرمز L1 . هذه الذاكرة يتم بناؤها داخل شريحة المعالج نفسه وتكون كميتها صغيرة حوالى 8 أو 16 كيلوبايت فى العادة .

المستوى الثانى من الذاكرة المخبأة Level 2 ويرمز لها بالرمز L2 ، ويتم بناؤها خارج شريحة المعالج وتكون امتداد لذاكرة المستوى الأول . أى أن البيانات تنتقل منها إلى ذاكرة المستوى الأول فى حالة القراءة ، ومن ذاكرة المستوى الأول إليها فى حالة الكتابة وليس للمستخدم أى سيطرة عليها . كمية هذه الذاكرة تكون كبيرة فى العادة حيث تبلغ 512 كيلوبايت .

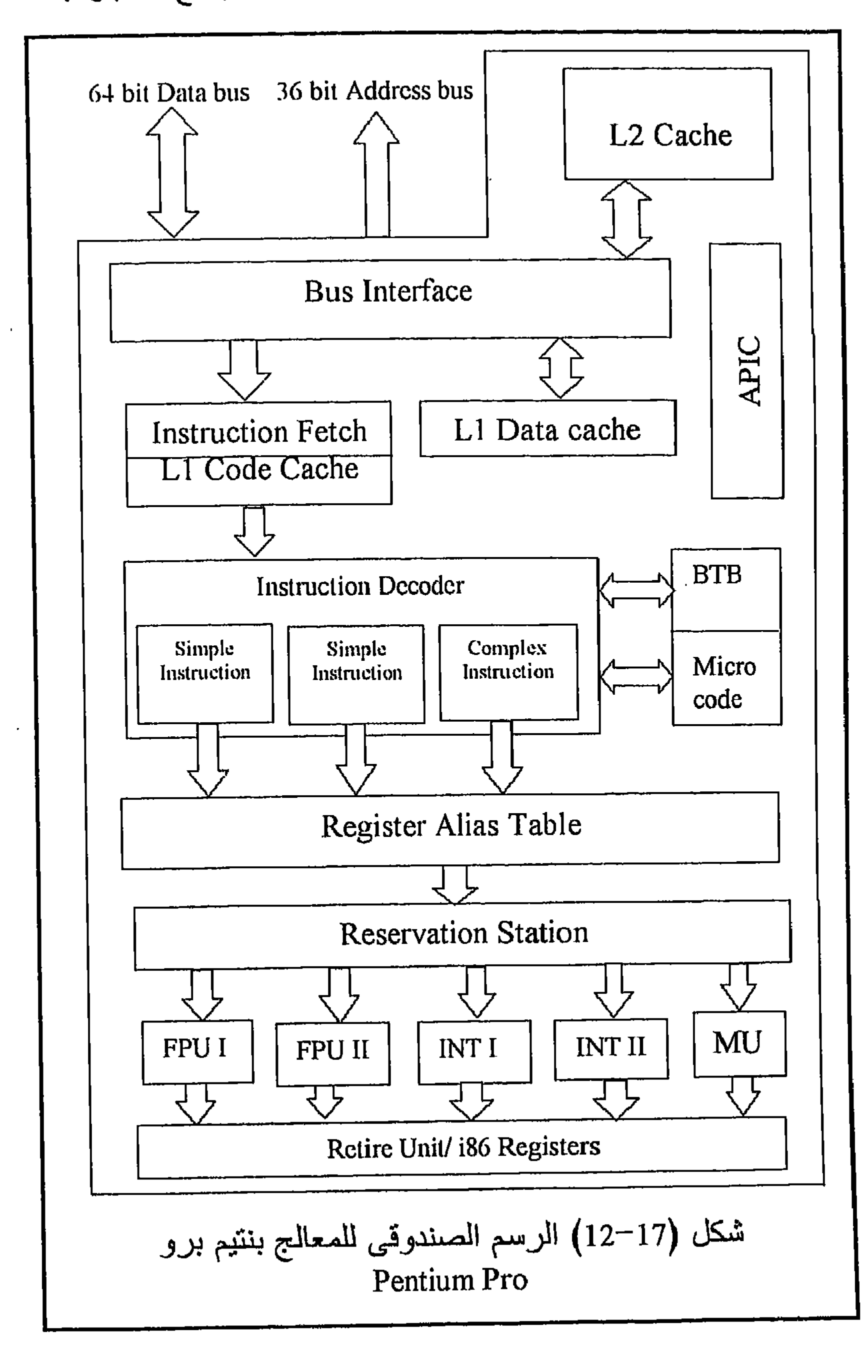
17-9 المعالج بنتيم برو
Pentium Pro Processor

لقد كان التطور التالى فى عائلة بنتيم هو المعالج بنتيم برو الذى تميز باحتوائه على كل ذاكرة المستوى الثانى المخبأة ، أى أنه يحتوى على 512 كيلوبايت ذاكرة مخبأة من المستوى الأول داخل نفس شريحة المعالج . لكى تتصور مدى كثافة المكونات على شريحة هذا المعالج فإنه يحتوى على خمسة ونصف مليون ترانزستور كلها مجمعة على شريحة تبلغ مساحتها 7.26 سم × 6.25 سم . هذه الشريحة تستهلك قدرة كهربية مقدارها 38 وات (12 أمبير تقريبا عند 3.3 فولت).

يحتوى المعالج بنتيم برو على 3 وحدات انسياب Pipeline كل منها تتكون من 12 مرحلة ، كما يحتوى على وحدة معالجة البيانات الحقيقية Floating Point Unit, FPU .

لقد صدر المعالج بنتيم برو فى شريحة شبكية متداخلة الأرجل Staggered Pin Grid Array, SPGA . شكل (17-12) يبين رسما صندوقيا لهذا المعالج . يتضح لنا من هذا الشكل أن مسار البيانات يتكون من 64 بت ، بينما مسار العناوين يتكون من 36 بت . هناك أيضا وحدة مواجهة المسارات Bus Interface Unit والتى تمثل حلقة الوصل بين المسارات الخارجية ووحدتى الذاكرة المخبأة

الداخلية ، حيث الوحدة الأولى تمثل وحدة ذاكرة مخبأة للبيانات ، والوحدة الثانيـــة تمثل ذاكرة مخبأة للأوامر وكل منهما من المستوى الأول وتبلغ 8 كيلوبايت .



شكل (17-12) الرسم الصندوقي للمعالج بنتيم برو
Pentium Pro

الجديد أيضا أن كل واحدة من وحدتى الذاكرة المخبأة السابقتين لها مسار بيانات خاص لقراءة البيانات ومسار آخر للكتابة بحيث يمكن القراءة والكتابة فى نفس الوقت من أى واحدة فى نفس نبضة التزامن . تمثل وحدة مواجهة المسارات أيضا حلقة الوصل بين المسارات الخارجية ووحدة الذاكرة المخبأة الثالثة .
تقوم وحدة إحضار الأوامر Instruction Fetch, IF بإحضار الأوامر من الذاكرة المخبأة للأوامر إلى مشفر الأوامر الذى يحتوى على 3 وحدات انسياب Pipelining تعمل على التوازى ، اثنان منها لتشفير الأوامر البسيطة ، والثالث لتشفير الأوامر المركبة CISC والتى تتطلب برمجيات قصيرة لتنفيذها .
يحتوى المعالج بنتيم برو على 40 مسجلا عاما تقوم وحدة جدول المسجلات Register Table المبينة فى الشكل بالتنسيق بينها وبين المسجلات العامة المعروفة من المعالجات السابقة . يوضح هذا الشكل أيضا احتواء هذا المعالج على خمس وحدات لتنفيذ الأوامر ، اثنان منها لتنفيذ الأوامر ذات البيانات الحقيقية FPUI و FPUII ، واثنان لتنفيذ الأوامر ذات البيانات الصحيحة INTI و INTII ووحدة مواجهة الذاكرة Memory Interface Unit, MIU . كل هذه الوحدات تعمل منفصلة وغير معتمدة على بعضها مما يمكنها من تنفيذ خمسة أوامر فى نفس الوقت على التوازى وفى نفس نبضة التزامن الواحدة . بالطبع لا يخلو الأمر من بعض الأوامر الشاذة التى تحتاج لمعاملات خاصة وهذه تحل مشاكلها فى وحدة العزل Retire unit والتى تحل فيها جميع مشاكل أوامر التفريع والقفز . آخر وحدة فى هذا الشكل هى وحدة المقاطعة المتقدمة القابلة للبرمجة Advanced Programmable Interrupt Controller, PIC والتى تستخدم أساسا لتنشيط عمليات المعالجة المتعددة multiprocessing باستخدام حتى 4 معالجات من هذا النوع دون الاحتياج لأى إضافات .
بذلك نكون قد وقفنا على جميع المعالجات بجميع أنواعها ، ونكون قد تأكدنا من أن فكرة المعالج الأساسية لم تتغير ابتداء من المعالجات 8 بت وانتهاء بالمعالجات 64 بت .

17-10 تمارين

1. ما هو مقدار الذاكرة التى يمكن أن يتعامل معها كل من المعالجات التالية :
   - المعالج 8086
   - المعالج 80186
   - المعالج 80286
   - المعالج 80386
   - المعالج 80486

• المعالج بنتيم
• المعالج بنتيم برو

2. كم عدد بتات مسار البيانات فى كل المعالجات السابقة ؟
3. اشرح نظام الذاكرة فى كل واحد من المعالجات السابقة وكيفية تنشيط البنكات المختلفة فى كل حالة ؟
4. اشرح نظام الإدخال والإخراج فى كل من المعالجات السابقة وكيفية تنشيط البنكات المختلفة فى كل حالة أيضا ؟
5. ما هى وظيفة الطرف $\overline{BS16}$ فى المعالج 80386 ؟
6. ما هى الذاكرة المخبأة cache memory ؟ وما هو أول معالج بدأ فى استخدامها ؟ وما مقدارها فى كل معالج من المعالجات التى استخدمتها ؟
7. اشرح المقصود بملف المسجلات Register file ؟ وما هو أول معالج بدأ فى استخدامه ؟ وما هو عدد المسجلات فى كل معالج من المعالجات التى استخدمت هذه الفكرة ؟
8. ما هو الفرق بين الذاكرة المخبأة وملف المسجلات ؟ وهل يمكن الاستغناء عن أى منهما على حساب الآخر ؟
9. اشرح فكرة الانسياب Pipelining ؟ وما هى المعالجات التى تستخدم هذه الفكرة ؟
10. ما هو المقصود بالحاسبات ذات القائمة المخفضة RISC ؟ وعلى أى شيء تم التخفيض ؟
11. هل هناك علاقة بين فكرة تخفيض الأوامر RISC والانسياب ؟
12. اذكر خصائص ومميزات وعيوب الحاسبات RISC ؟
13. ما هو الفرق بين المعالجين بنتيم وبنتيم برو؟
14. إذا طلب منك تصميم دائرة تتحكم فى إشارة مرور فى تقاطع معين ، فأى المعالجات التى درستها تختار ؟
15. ما رأيك فى الاستغناء عن دراسة المعالجات 8 بت والاكتفاء بدراسة آخر الأجيال منها وهو المعالج بنتيم برو مثلا ؟

الملحق الأول ... 1

الحساب الثنائى
Binary Arithmatic

مقدمة

لقد رأينا فى الفصل السابع كيفية إجراء الجمع الثنائى والطرح الثنائى باختصار شديد وقد كان التركيز الكلى على الدوائر التى تفى بهذه الأغراض ، ولكن تبقى حتى الآن بعض الأسئلة التى مازالت تحتاج إلى إيضاح ومنها كيف يتعامل المعالج مع الأرقام السالبة ؟ وكيف يميز بين رقم سالب وآخر موجب ؟ وأنا كمبرمج كيف أعرف إذا كانت النتيجة سالبة أو موجبة ؟ وماذا عن عمليات الضرب والقسمة ؟ كل هذه الأسئلة سنجيب عنها فى هذا الملحق إن شاء الله .

عمليات الجمع والطرح

انظر إلى عمليات الجمع الثنائى التالية :

BCD	عشرى	BCD	عشرى
0010 0101	25	0101 0101	55
0011 0001 +	31 +	0100 0100 +	44 +
0101 0110	56	1001 1001	99

نلاحظ فى هذين المثالين التطابق التام فى نتيجة الجمع فى كلا النظامين العشرى والعشرى المكود ثنائيا BCD . الآن انظر إلى عمليتى الجمع التاليتين :

DCB	عشرى	BCD	عشرى
0010 1001	29	0101 0101	55
0010 0010 +	22 +	0011 1000 +	38 +
0100 1011	51	1000 1101	93

نلاحظ فى المثالين السابقين عدم التطابق بين الصورة العشرية والصورة الثنائيــة ولكن الظريف فى الأمر أننا يمكن أن نحصل على التطابق المطلـــوب بمجــرد إضافة الرقم 6 (0110) إلى النتيجة كما يلى :

عشرى	BCD	عشرى	DCB
55	0101 0101	29	1001 0010
+ 38	+ 0011 1000	+ 22	+ 0010 0010
93	(خطأ) 1000 1101	51	(خطأ)0100 1011
	+ 0110		+ 0110
	(صح) 1001 0011		(صح)0101 0001

من ذلك نخرج بنتيجة مهمة وهى أنه فى حالة جمع رقمين كل منهما 4 بتات فإنه إذا كانت النتيجة أقل من أو تساوى 9 فإن هذه النتيجة تكون متطابقة مع النظام العشرى ولا تحتاج لعملية ضبط adjust. أما إذا كانت النتيجة أكبر من 9 أو كان هناك حمل من الخانة الثالثة (الأخيرة) فإنه للحصول على النتيجة فى الصورة العشرية فلابد من عملية ضبط باضافة الرقم 6 (0110) للنتيجة . لزيادة التأكيد على ذلك انظر للأمثلة التالية :

3	0011	4	0100	9	1001
+ 5	+ 0101	+ 9	+ 1001	+ 9	+ 1001
8	1000	13	1101	18	10010
			+ 0110		+0110
			1 0011		11000

أما فى حالة جمع رقمين كل منهما 8 بتات فإن نفس النتيجة السابقة تطبق علـــى كل نصف من النتيجة على حده . أى أنه إذا كان النصف الأول من النتيجة أكــبر من 9 أو كان هناك حمل من الخانة الثالثة إلى الخانة الرابعة (أى أن علم الحمل البينى HCF يساوى واحدا) فإنه يجب إضافة الرقم 6 (0110) إلى النتيجة كعمليــة ضبط . وأما إذا كان النصف الثانى من النتيجة أكبر من 9 أو حصل حمل مـــن الخانة السابعة (أى علم الحمل يساوى واحد) فإنه يجب إضافة الرقم 60H (0110 0000) إلى النتيجة كعملية ضبط للحصول على النتيجة فى الصورة العشـــرية . الأمثلة التالية توضح ذلك :

```
حصل حمل من الخانة الثالثة إلى الرابعة      1001 0101        59
لذلك لزم إضافة 0110 إلى النتيجة.           0011 1001 +      39 +
                                          ----------       ----
                                          1001 0010        98
                                               0110 +
                                          ----------
                                          1001 1000
```

```
                                          1000 1000        88
                                          0100 1001 +      49 +
                                          ----------       ----
حصل حمل من الخانة الثالثة إلى الرابعة      1101 0001        137
لذلك لزم إضافة 0110 إلى التيجة .               0110 +
                                          ----------
النصف الأخير من النتيجة أكبر من 9          1101 0111
لزم إضافة 0110 0000 إلى التيجة .           0110 0000 +
                                          ----------
                                        1 0011 0111
```

إن عملية إضافة الرقم 6 أو 60H إلى النتيجة تسمى عملية الضبط العشرى وهذه العملية يقوم بها الكثير من المعالجات بناء على الأمر DAA والذى يعنى Dicimal Adjust accumulator after Addition أى ضبط المركم عشريا بعد عملية الجمع .

يمكن إجراء نفس عملية الضبط السابقة بعد عملية الطرح ، الإختلاف هو أننا نطرح الرقم 6 من النتيجة إذا كان النصف الأول منها أكبر من 9 أو كان علم الحمل البينى HC يساوى واحدا ، ونطرح الرقم 60H من النتيجة إذا كان النصف الثانى منها أكبر من 9 أو كان علم الحمل CF يساوى واحدا ، الأمثلة التالية توضح ذلك :

```
                                                  1000 0110     86
                                                  0101 0111 -   57 -
                                                  ----------    ----
النصف الأول من النتيجة أكبر من 9 لذلك لزم          0010 1111     29
طرح الرقم 6 منها .                                      0110 -
                                                  ----------
                                                  0010 1001
```

1001 0100	49
0111 0010 −	72 −
1101 0111	77
0110 0000 −	
0111 0111	

النصف الثانى من النتيجة أكبر من 9 لذلك لزم طرح الرقم 60H منها .

لننسى كون النتيجة يجب أن تكون سالبة أو موجبة الآن ونعلم أن عملية طـــرح الرقم 6 أو 60H من النتيجة يقوم بها الكثير من المعالجات بناء على الأمـر DAS أى Decimal Adjust accumulator after Subtraction والتى تعنى ضبط المركم عشريا بعد عملية الطرح .

تمثيل الأرقام السالبة فى النظام الثنائى

فى النظام العشرى نستطيع تمييز الأرقام السالبة بوضع إشارة (−) قبل الرقم كما نستطيع تمييز الأرقام الموجبة بوضع الإشارة (+) أمامه . أما فى النظام الثنائى حيث الواحد والصفر هما الشكلان الوحيدان المتاحان للاستخدام فلابد وأن يكـون هناك وسيلة لتمثيل الأرقام السالبة والموجبة . ولقد تم التعارف علـى أن تكـون آخر بت فى الرقم تمثل إشارة ذلك الرقم وتم التعارف أيضا على أنه إذا كـانت آخر بت تساوى صفرا فإن ذلك الرقم يكون موجبا وإذا كانت آخر بت تسـاوى واحدا فإن ذلك الرقم يكون سالبا . وإليك بعض الأرقام السالبة والموجبة علـى حسب التعارف السابق :

خانة الإشارة

+7	0	0000111
+46	0	0101110
+64	0	1000000
-12	1	1110100
-46	1	1010010

المتمم الثنائى

لتسهيل العمليات الحسابية ، وبالذات عمليات الطرح , فى ظـل هـذا التعـارف السابق فقد تم استخدام نظام المتمم الثنائى لتمثيل الأرقام السالبة . المتمم الثنـائى لأى رقم ثنائى يمكن الحصول عليه بعكس كل بت فى الرقم (جعل الواحد صفـرا

والصفر واحدا) ثم إضافة واحد لنتيجة هذا العكس . هذا المتمم الثنائى للرقم يمثل الرقم سالبا . الأمثلة التالية توضح ذلك :

- الرقم 7+ يمثل ثنائيا بالرقم 00000111 حيث نلاحظ أن آخر بت تساوى صفرا دلالة على أن هذا الرقم موجب . اذن كيف نمثل الرقم 7- ؟
- للحصول على الرقم 7- الثنائى نتبع الخطوات التالية :

الرقم 7+ هو	00000111
اعكس الرقم	11111000
أضف واحدا	11111001

هذه النتيجة الأخيرة(11111001) تمثل الرقم 7- فى النظام الثنائى .

لاحظ أنه فى ظل تخصيص الخانة الأخيرة للإشارة أصبحت قيمة الرقم ممثلة فقط بسبعة (بتات) ، أى أن قيمة الرقم قد نقصت حيث كانت قيمته تتراوح بين الصفر و 255 قبل اعتبار الخانة الأخيرة كخانة إشارة ، أما فى ظل اعتبار الخانة الأخيرة كخانة إشارة فلقد أصبحت قيمة الرقم تتراوح بين الصفر و +127 للأرقام الموجبة (البت الثامنة صفر) ، وتتراوح قيمة الرقم بين -1 و -128 للأرقام السالبة (بت الإشارة واحد) . وعلى ذلك يمكن كتابة هذه الأرقام كما يلى:

+127	01111111
+126	01111110
+125	01111101
....	
+1	00000001
0	00000000
-1	11111111
-2	11111110
....	
-127	10000001
-128	10000000

افترض أن أمامك رقما وتريد معرفة قيمة هذا الرقم وهل هو سالب أم موجب ؟ فى هذه الحالة عليك أولا النظر إلى خانة الإشارة فاذا كانت صفرا فإن هذا الرقم موجب وتحدد قيمته بباقى الخانات السبعة . أما إذا كانت خانة الإشارة تحتوى واحدا فإن ذلك يعنى أن هذا الرقم سالبا وتحدد قيمته بعد حساب المتمم الثنائى للرقم . كمثال على ذلك افترض أن لديك الرقم 11111001 ، هذا الرقم سالب لأن آخر خانة تساوى واحدا ، وللحصول على قيمة هذا الرقم نحسب المتمم الثنائى له كالتالى :

الرقم	11111001
اعكس	00000110
أضف واحدا	00000111

إذن هذا الرقم هو 7-

إليك الآن بعض الأمثلة على عمليات الجمع والطرح للأرقام ذات الإشارة :

```
  +13      00001101
+ +9       00001001
  +22      00010110
```

أخر بت فى النتيجة تساوى صفرا لذلك فالنتيجة موجبة وتساوى 22 .

لاحظ أننا نتعامل فى النظام الثنائى وليس النظام الستعشرى .

```
  +13      00001101
+ -9       11110111     المتمم الثنائى للرقم 9
  +4     1 00000100
```

تم إهمال الحمل الناتج ، وطالما أن أخر بت فى النتيجة تساوى صفرا فالنتيجة موجبة وتساوى 4+ .

```
  +9       00001001
+ -13      11110011     المتمم الثنائى للرقم 13
  -4       11111100
```

أخر بت تساوى واحد فالنتيجة سالبة ، خذ المتمم الثنائى للنتيجة وهو

00000100 تحصل على النتيجة النهائية ، لذلك فالنتيجة النهائية تساوى 4- .

```
  -13      11110011
+ -9       11110111
  -22    1 11101010
           00010110
```

إهمل الحمل , خذ المتمم الثنائى للنتيجة وهو 00010110 لذلك فالنتيجة تساوى 22-

فى الأمثلة السابقة كنا نتعامل فى النظام العشرى أو النظام الثنائى ، ماذا سيكون الموقف عند التعامل مع النظام الستعشرى الشائع الاستخدام فى الكثير من أنظمة المعالج . إليك بعض الأمثلة التى توضح ذلك :

00010011	+13H
11110111	- 9H +
1 00001010	+0AH

إهمل الحمل والنتيجة هى 0A+ لأن آخر بت فى النتيجة تساوى صفرا .

	00001001	+ 9H
المتمم الثنائى للرقم 13H (00010011)	11101101	−13H +
آخر بت تساوى 1 فالنتيجة سالبة , خذ المتمم الثنائى	11110110	-0AH
لذلك فالنتيجة هى 0A- .	00001010	

من ذلك نرى أن عملية تمثيل الأرقام السالبة بالمتمم الثنائى للرقم هى نفسها فى أى من النظامين العشرى أو الستعشرى ، كل ما هناك هو ملاحظة الفرق فى كيفية وضع الرقم فى صورته الثنائية . إن الأمثلة السابقة توضح لنا أن عملية الطرح ما هى إلا عملية جمع للمطروح منه مع المتمم الثنائى للمطروح وذلك ما تقوم به عادة دوائر المعالج التى تقوم بتنفيذ عملية الطرح كما رأينا فى الفصل السابع .

الضرب والقسمة ثنائيا

يمكن إجراء الضرب الثنائى بأكثر من طريقة أولها هى طريقة الضرب فى النظام العشرى والتى نعرفها جميعا منذ الصغر حيث يتم ضرب الخانة الأولى من المضروب فى المضروب فيه وتدون النتيجة ، ثم يتم ضرب الخانة الثانية من المضروب فى المضروب فيه وتدون النتيجة تحت النتيجة السابقة ولكن تزاح لليسار بمقدار خانة ، وهكذا يتم ضرب جميع خانات المضروب فى المضروب فيه وفى كل مرة تدون النتيجة تحت النتيجة السابقة بعد إزاحتها بمقدار خانة واحدة ، وفى النهاية يتم جمع جميع النتائج السابقة كما يلى :

```
المضروب فيه  1011        11
المضروب     1001       x 9
          -------     -----
             1011        99
            0000
           0000
          1011
        -------
        1100011
```

من الطرق الأخرى للضرب استخدام طريقة الجمع المتكرر حيث فى المثال السابق مثلا نقوم بجمع العدد 11مع نفسه 9 مرات . من عيوب هذه الطريقة أنها تكون بطيئة بالذات فى حالة ضرب الأرقام الكبيرة ، فمثلا فى حالة ضرب

الرقمين 165 فى 99 سنقوم بجمع الرقم 165 مع نفسه 99 مـــرة وذلـــك بـــالطبع يتطلب الكثير من الوقت .
فى المثال السابق (11x9) نرى أن نتيجة ضرب أى خانة من المضـــروب فـــى المضروب فيه تكون إما صفرا أو المضروب فيه نفسه حيث أن هذه الخانة تكـون إما صفرا أو واحدا . يمكن الاستفادة من ذلك فى استنتاج طريقة أخرى للضـــرب أسرع وأنسب مـــن الطريقتيـــن الســـابقتين كمـــا ســـنرى فـــى المثـــال التـــالى (13x13=169) :

1101	
× 1101	
1101	طالما أن البت الأولى من المضروب 1 كتبنا المضروب فيه كما هو.
01101	ثم أزحنا النتيجة السابقة لليمين بمقدار خانة واحدة .
001101	طالما أن البت الثانية من المضروب 0 لم نفعل شى فقط أزحنا النتيجة السابقة لليمين بمقدار خانة .
001101	طالما أن الخانة الثالثة من المضروب 1 جمعنا المضروب فيه
+ 1101	على النتيجة السابقة فنحصل على النتيجة التالية :
1000001	
01000001	ثم أزحنا النتيجة السابقة ناحية اليمين بمقدار خانة .
01000001	طالما أن البت الرابعة فى المضروب 1 نجمع المضروب فيه
+ 1101	على النتيجة السابقة فنحصل على النتيجة النهائية التالية :
10101001	169

أى أن هذه الطريقة تتلخص فى أننا نضرب الخانة الأولى من المضـــروب فـــى المضروب فيه ثم نزيح النتيجة ناحية اليمين بمقدار خانـــة ، وإذا كـــانت الخانـــة الثانية صفرا فلا نعمل شيئا سوى الإزاحة لليمين بمقدار خانة ، وإذا كانت واحـــدا نجمع المضروب فيه مع النتيجة السابقة ثم نزيح نتيجة الجمع السابقة خانة وننظر فى الخانة الثالثة من المضروب ، وهكذا . نلاحظ أن هذه الطريقة أسرع بكثـــير من الطرق السابقة كما نلاحظ أن النتيجة النهائية للضرب تشغل عددا من البتـــات يساوى مجموع عدد بتات المضروب والمضروب فيه .
لإجراء عملية القسمة الثنائية نتبع نفس الخطوات التى اتبعناها فى إجراء عمليات الجمع الثنائى ، سوى أننا نستخدم الطرح المتكرر بدلا من الجمع المتكرر كما فى حالة الجمع ، أو نستخدم الطرح ثم الإزاحة ناحية اليسار بدلا مـــن الجمـــع ثـــم الإزاحة ناحية اليمين كما فى حالة الضرب .

مراجع الكتاب

1. The MFA microcomputer training kit manuals , 1986.
2. Heathkit manual for the microprocessor trainer model ET-3400, 1981.
3. Ramesh S. Gaonkar, "Microprocessor architecture, programming and applications with the 8085/8080A." Wiley Eastern Limited, 1986.
4. Douglas V. Hall "Microprocessors and digital systems" Mcgraw Hill Book company,1983.
5. Kathe Spracklen "Z-80 and 8080 assembly language programming" Hayden Book Company, Inc, 1979.
6. Charles K. Adams "Master handbook of microprocessor chips" TAB Books Inc. 1981.
7. David F. Stout "Microprocessor applications handbook" McGraw Hill Book Company, 1985.
8. James W. Coffern "Z-80 Applications" ترجمة الدار العربية للعلوم 1988.
9. Lance A. Lventhal "6800 assembly language programming" Osborne & Associates Inc. 1978.
10. Joseph J. Carr "Microprocessor Interfacing" TAB Books Inc. 1982.
11. Frank P. Tedeschi & Robert Colen "101 projects for the Z-80" TAB Books Inc. 1983.
12. Hermann Schmid "Electronic analog/digital conversions" Van Nostrand Reinhold Company, 1970.
13. Micro-tech Publication "Microprocessor Data Hand Book" 1991
14. Michael Thorne "Programming the 8086/8088 for the IBM PC and Compatables" The Benjamin P. Company 1986
15. د. عوض منصور "البرمجة بلغة التجميع" 1991 الطبعة الأولى.
16. Barry B. Brey "The Intel Microprocessors 8086, 80186, 80286, 80386, 80486" Maxwell International 1991 .
17. Hans-Peter Messmer "The Indispensable PC Hardware Book" Addison- Wesley 1997 .
18. Daniel Tabak "Advanced Microprocessors" McGraw Hill Inc. 1995

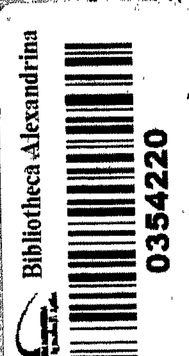
المعالجات الدقيقة

Microprocessors

تأليف / د. محمد إبراهيم العدوي

هذا الكتاب

ما هو الميكروبروسيسور (المعالج الدقيق)؟ أين يقع المعالج فى الميكروكومبيوتر؟ كيف يمكن برمجته؟ كيف يمكن توصيله مع الذاكرة؟ كيف يمكن إدخال بيانات إليه، وكيف يمكن إخراج بيانات منه؟ كيف يمكن استخدام المعالج كعنصر أساسى فى دوائر التحكم مثل التحكم فى درجات الحرارة والضغط وإشارات المرور وغير ذلك؟ ما هو الميكروكومبيوتر ذو الكارت الواحد؟ ما هى البرامج الفرعية؟ كيف يمكن مقاطعة الميكروبروسيسور؟ كل هذه الأسئلة وأكثر يجيب عنها هذا الكتاب فى أسلوب سهل وأفكار متدرجة خالية من التعقيد ومن خلال أمثلة وتمارين كثيرة مطبقة على أشهر المعالجات ذات 8 بت وهى المعالج intel 8085 والمعالج Z80 وكذلك أشهر المعالجات ذات 16 بت وهو المعالج intel 8086. ولم يقف الكتاب عند هذا الحد بل قدم المعالجات الأخرى مثل المعالجات 80186 و 80286 و 80386 و 80486 وحتى الأجيال المتأخرة من عائلة بنتيم...



IHCI

...TIONAL HOUSE FOR CULTURAL INVESTMENTS

CAIRO - EGYPT

ISBN : 977-282-076-5